Dissertation zur Erlangung des akademischen Grades des Doktors der Ingenieurwissenschaften

Verwendung von *Hover Detection* zur Verbesserung der Texteingabe auf Smartphones

Dipl.-Inf. Frederic Pollmann

28. Mai 2017

Dissertationskolloquium am 16. Oktober 2017

vorgelegt im Fachbereich 3 – Mathematik & Informatik
Universität Bremen

1. Gutachter: Prof. Dr. Rainer Malaka
2. Gutachter: Prof. Dr. Udo Frese

Für meine Familie

Für Birgit

Zusammenfassung

Mobile Geräte wie Smartphones sind für viele Menschen aus dem Alltag nicht mehr wegzudenken. Jedoch fällt nicht allen Interaktion mit einem Smartphone leicht, gerade bei der Texteingabe. Der geringe Platz auf dem Bildschirm dieser in der Hand gehaltenen Geräte limitiert der Größe der Steuerelemente auf der Nutzeroberfläche. Dieses Problem verstärkt sich bei gleichzeitiger Anzeige einer Vielzahl von Steuerelementen, zum Beispiel bei einer Bildschirmtastatur. Vor allem Nutzern mit Einschränkungen wie verminderter Sehfähigkeit oder Feinmotorik kann es schwer fallen, diese Geräte zu verwenden. Dies schließt sie von Teilen unseres modernen Soziallebens aus, das auf der Möglichkeit ständiger Kommunikation über Text- oder Bildnachrichten aufbaut.

In dieser Arbeit wurde die Eignung von *Hover Detection* (engl. für Schwebeerkennung) zur Verbesserung der Nutzerfreundlichkeit bei der Texteingabe auf dem Smartphone untersucht. In aufeinander aufbauenden Experimenten wurde die Position des schwebenden Fingers genutzt, um die Darstellung der Nutzeroberfläche kontextsensitiv zu vergrößern oder um visuelles oder akustisches Feedback zur Position des Fingers über der Tastatur zu geben. In Tests mit älteren Nutzern wurde das visuelle Feedback anfänglich gut aufgenommen. Leider stellte sich heraus, dass die vergleichsweise hohe Latenz der Hover Detection von 250 ms eine positive Auswirkung auf die Nutzerfreundlichkeit verhinderte. Dieses Ergebnis wurde in einem Versuch mit jungen Nutzern bestätigt, die ebenfalls keinen Vorteil aus der Unterstützung durch Hover Detection ziehen konnten. Es konnte jedoch gezeigt werden, dass eine dauerhafte Vergrößerung der Tasten die Nutzerfreundlichkeit für Senioren verbessern konnte. Nutzer mit eingeschränkter Sehfähigkeit nahmen die Idee kontextsensitiver Vergrößerung anfänglich ebenfalls positiv auf, konnten Hover Detection aber aufgrund des fehlenden haptischen Feedbacks nicht gut nutzen. Akustisches Feedback konnte aus dem gleichen Grund nicht zur Verbesserung der Nutzererfahrung beitragen. Eine zuverlässige Nutzung von Hover Detection war ohne hinreichende Sehfähigkeit nicht möglich.

In dieser Forschungsarbeit konnte gezeigt werden, dass unterstützende Funktionen wie selektive Vergrößerung der grafischen Schnittstelle auf Smartphones den Nutzern helfen können. Dies trifft allerdings nur zu, wenn die technischen Rahmenbedingungen den Eingabevorgang hinreichend unterstützen. Im vorliegenden Fall war dies aufgrund zu hoher Latenzen nicht der Fall. Sollte die für Hover Detection verfügbare Hardware sich in Zukunft verbessern, könnten diese Ergebnisse dieser Forschungsarbeit wieder relevant werden.

Abstract

Interaction with smartphones can be challenging for some users, especially with regards to text entry. In these handheld devices the available screen space limits the size of the user interfaces elements. This problem is excerbated when a lot of UI elements has to be displayed simultaneously, for example in a on-screen keyboard. Especially users with limitations like decreased vision or motor control can have a hard time using these devices, effectively excluding them from a part of modern social life.

In this work we evaluated if *hover detection* can be used to improve usability for text entry on a smartphone. In several experiments the position of the hovering finger was used to selectively enlarge the UI, to provide visual location feedback on the keyboard or to offer audio assistance. When testing with elderly users, the visual feedback was positively received. Unfortunately the comparatively high latency of the hover detection (about 250 ms) negated any gains in usability. This result was confirmed in tests with young users, who also did not benefit from the hover detection. Most usability gains for elderly users were made by introducing a keyboard layout with larger keys which stayed at that size, regardless of hover position. Visually impaired users liked the idea of a context sensitive magnification as well, but hover detection was not really usable to its inherent lack of haptic feedback. Acoustic feedback did not produce a better user experience for the same reason. Reliable use of hover detection was just not possible without adequate levels of vision.

This research showed that assistive technologies on smartphones like selective magnification of the user interface can help users, but only when technical parameters are sufficient for the input process. In this case hover detection allowed us to implement visual, haptic and audio feedback based on the hover position of the finger as a proof of concept. Unfortunately high latency only allowed us to show qualitative improvement, not quantitive. Further improvements in hover detection hardware may make this research relevant again, though.

Inhaltsverzeichnis

1 Einleitung

Wir leben im Informationszeitalter. Was bedeutet das? Informationen sind nun als Rohstoffe und Waren essentiell. Beinahe jede Person auf diesem Planeten hat Zugang zu einem wie auch immer gearteten Computer. Die meisten dieser Geräte verfügen über die Möglichkeit des elektronischen Datenaustausches, sei es nun klassisch über das Internet oder auch nur über das Mobilfunknetz mit der Möglichkeit Textnachrichten zu verschicken. Auf diese Weise kann sich Kommunikation und die Verbreitung von Wissen beinahe verzögerungsfrei und global abspielen, die alten physikalischen Einschränkungen durch Entfernung und Besitz sind teilweise bedeutungslos geworden.

Diese Revolution der Kommunikation findet in großen Teilen mit Smartphones, Tablets und anderen mobilen Computern statt. Sie sind inzwischen fast so leistungsfähig wie klassische Desktop- oder Notebook-Computer. Dabei können sie aufgrund ihres geringes Gewichts und der handlichen Größe beinahe immer mitgeführt werden und ermöglichen jederzeit den Zugriff auf den größten Informationsschatz, den die Menschheit je hatte: Das *World Wide Web*. Dafür ist nach wie vor die Eingabe von Begriffen notwendig, auch wenn sich die Art der Eingabe zusammen mit den immer schneller und kleiner werdenden Geräten weiter entwickelt hat und immer besser an die Bedürfnisse des Menschen angepasst wurde. Dies ist dem wissenschaftlichen Feld der Mensch-Maschine-Interaktion zu verdanken, oft abgekürzt als *HCI* (vom englischen „Human-Computer Interaction“), das sich mit den Gestaltungsregeln für solche Interaktionsvorgänge beschäftigt.

Prägnant ausgedrückt: HCI untersucht, wie die Steuerung von Maschinen durch den Menschen so einfach und angenehm wie möglich zu gestalten ist.

Ein wichtiger Aspekt bei der Interaktion mit Computern war immer schon die Eingabe der zu verarbeitenden Daten. Beginnend bei den in Großrechnern verwendeten Lochkarten, die ohne tiefgehende Vorbildung für einen Menschen nicht verständlich waren, sind wir heute bei intuitiv nutzbaren Mechanismen wie Handschrifterkennung oder Spracheingabe angekommen. Lochkarten enthielten direkt kodierte Anweisungen und Daten für die Maschine, bevor sie durch die Eingabe textbasierter Befehle in einer Kommandozeile abgelöst wurden. Diese orientieren sich zwar meist schon an menschlicher (meist englischer) Sprache, erzwingen jedoch eine spezifische Syntax und Verwendung der Befehle. Obwohl diese Systeme ohne Vorbildung kaum zu verwenden sind, gelang mit ihnen der Sprung des Computers aus den Rechenzentren in den persönlichen Lebensbereich der Menschen. Diese Geräteklasse wird daher bis heute als „Personal Computer“ (abgekürzt: PC) bezeichnet.

Die einfache Nutzung durch „jedermann“ erforderte ein Umdenken bei der Bedienung dieser Computersysteme. Für den *Xerox Alto* (1973) wurde eine erste grafische Schnittstelle auf Basis der Elemente *Window*, *Icon*, *Menu* und *Pointer* (*WIMP*) entwickelt, die auf eine Bedienung mit der neu erfundenen Maus setzte. Mit dem *Apple Macintosh* (ab 1984) feierte dieses Konzept seinen Durchbruch und war spätestens seit der Verwendung auf PCs mit *Microsoft Windows* (ab 1985) der neue Standard für die Bedienung eines Computers. Die Steuerung des Systems mittels einer Kommandozeile wurde durch dieses neuer Paradigma fast völlig verdrängt. Für die Nutzung des Systems zur elektronischen Datenverarbeitung spielt Text jedoch nach wie vor eine zentrale Rolle. Zur Texteingabe dient bei PCs bis heute standardmäßig eine physikalische Tastatur. Selbst mit zunehmender Verkleinerung und Mobilisierung von Computer-Systemen wurde die Tastatur beibehalten und in ihrer Größe entsprechend reduziert.

Erst das 2007 von Apple Inc. vorgestellte *iPhone* führte mit seinem berührungsempfindlichen Bildschirm die direkte Interaktion ohne weitere Hilfsmittel ein. Dieser nimmt fast die gesamte Front des Geräts ein und kann Berührungen mit dem Finger erkennen; meist ist sogar die gleichzeitige Erkennung mehrerer Finger möglich. Diese Technik ermöglicht es, auf eine fest verbaute Tastatur zu verzichten und sie nur bei Bedarf auf dem Bildschirm anzuzeigen. Da die Tastatur nun nur noch als Software existiert und keine dedizierte Hardware mehr verwendet, spricht man von einem „Soft-Keyboard“ oder aufgrund der Darstellung von einer „Bildschirmtastatur“.

Diese Doppelnutzung des Bildschirms für Ein- und Ausgabe ermöglicht die Herstellung kompakterer Geräte, die durch den Wegfall mechanischer Einzelteile noch zuverlässiger und langlebiger sein können. Sie sind als Begleiter im Alltag ausgelegt und finden in einer Hosentasche Platz. Für die auf dem Bildschirm dargestellte Tastatur ist die geringe Größe jedoch ein Nachteil, da die dargestellten Tasten schwieriger zu treffen sind. Größere Bildschirme lassen sich zwar gut als Marketinginstrument nutzen, gleichzeitig nimmt jedoch die Portabilität ab und auch eine Einhandnutzung wird immer schwerer.

Trotz der rasanten Entwicklung der Technologie wird uns die Forschung an verbesserten Texteingabe-Lösungen als Konstante der Mensch-Computer-Interaktion wohl noch eine Weile erhalten bleiben. Auch die Berücksichtigung eventueller körperlicher Einschränkungen auf Seiten des Nutzers, zum Beispiel eine verringerte Sehfähigkeit, ist wichtig für eine einfache und intuitive Nutzung dieser zentralen Interaktion.

Neben komplett neuen Tastatur-Layouts und erweiterten Eingabemechanismen wie zum Beispiel Wischen statt Tippen sind es vor allem Vorhersage- und Fehlerkorrekturmechanismen, welche heute die Nutzung von Bildschirmtastaturen einfacher machen sollen. Neben den rein algorithmischen Lösungen gibt es manchmal auch neue technische Möglichkeiten, die das Potential haben Texteingabe zu verbessern. Neue Sensoren ermöglichen die Erfassung zusätzlicher Informationen über den Nutzer. Schwebeerkennung ist eine solche Erweiterung der bereits angesprochenen berührungsempfindlichen Bildschirme. Sie ermöglicht, die Position eines Fingers bereits im Schwebezustand über dem Bildschirm feststellen, bevor eine Berührung stattfindet. Nachdem einige auf

dem Massenmarkt verfügbare Mobiltelefone diese Technik unterstützen, soll mit der vorliegenden Dissertation die Frage beantwortet werden:

> Kann mit Hilfe von Schwebeerkennung die Texteingabe bei Nutzung einer Bildschirmtastatur verbessert werden?

Zu diesem Zweck wurden Experimente durchgeführt, welche die Schwebeerkennung an sich und im Kontext der mobilen Texteingabe untersuchten. Mit älteren Nutzern und Nutzern mit eingeschränkten Sehfertigkeiten wurde visuelles Feedback in Form von Vergrößerung und zusätzlichen Anzeigebereichen untersucht. Speziell mit Nutzern mit eingeschränkten Sehfertigkeiten wurde auch akustisches Feedback bei der Nutzung einer Bildschirmtastatur erprobt. Im Ergebnis zeigte sich allerdings, dass bei beiden Zielgruppen aufgrund technischer Probleme der Schwebeerkennung derzeit noch kein nennenswerter praktischer Vorteil bei der Texteingabe zu erzielen ist.

Inhalt und Gliederung dieser Arbeit Im Folgenden wird in Kapitel 2 „Stand der Wissenschaft“ zunächst ein Überblick über die Forschung gegeben, die für das Verständnis der durchgeführten Experimente und ihrer Ergebnisse notwendig ist. Besondere Schwerpunkte liegen dabei auf der „schwebenden Interaktion“ sowie dem Bereich Texteingabe. Aufbauend darauf werden in Kapitel 3 „Einsatz und Bewertung von Hover Detection auf Mobilgeräten“ die Herangehensweise, Prototypen und durchgeführten Experimente vorgestellt. Eine Auswertung und Interpretation der Ergebnisse dient in 4 „Diskussion und Ausblick“ als Grundlage für Empfehlungen zur weiteren theoretischen und praktischen Forschung in diesem Bereich. Abschließend wird dort auch das gewählte Vorgehen sowie der eigenen wissenschaftliche Beitrag kritisch hinterfragt. Für den technisch interessierten Leser werden in Anhang A noch einige Implementierungsdetails beschrieben, die für das Verständnis der Arbeit jedoch nicht notwendig sind.

Wahl der Sprache für Fachbegriffe Die Disziplin der Informatik ist stark mit dem englischsprachigen Raum verknüpft. Für viele englische Fachbegriffe gibt es keine passenden oder allgemein akzeptierte Übersetzungen. Daher werden einige Begriffe im Folgenden bewusst in der englischen Variante verwendet werden. Bei erster Nutzung eines solches Fremdwortes wird in der Regel aber eine Übersetzung sowie eine kurze Erklärung gegeben.

Nachdem dies nun geklärt ist, der wichtigste Begriff vorweg: Statt „Schwebeerkennung“ wird in den weiteren Kapiteln die Rede von „Hover Detection“ sein.

2 Stand der Wissenschaft

Die in der Einleitung beschriebene Forschungsfrage knüpft an verschiedene existierende wissenschaftliche Disziplinen an. Primär ist dies das Feld der Mensch-Maschine-Interaktion, in dem die einzelnen Bereiche wie die Hover-Interaktion, grafische Darstellung von Oberflächen oder die Texteingabe zu finden sind. Auch die Physik ist bei der technischen Umsetzung der Hover Detection beteiligt, die zugrundeliegenden Phänomene werden ebenfalls vorgestellt.

2.1 Mensch-Computer-Interaktion

Der Begriff „Mensch-Computer-Interaktion“ (engl. *human-computer interaction*, *HCI*) bezeichnet ein interdisziplinäres wissenschaftliches Feld, das sich mit der Nutzung von Computern durch Menschen beschäftigt.

Die *Special Interest Group on Computer-Human Interaction* der *Association for Computing Machinery* definiert das Feld so:

> Human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them. (Hewett u. a. 1996)

Abgedeckt sind dadurch also der Entwurf, die Bewertung und die Umsetzung von Computersystemen, die interaktiv von Menschen verwendet werden sollen, sowie die Untersuchung damit verbundener Effekte. Dabei kann ein „interaktives System“ viele Formen annehmen, sowohl eine Waschmaschine, ein Kraftfahrzeug oder ein Desktop-Computer oder Mobiltelefon fallen darunter.

Aufgrund der vielfältigen Anforderungen handelt es sich bei Mensch-Computer-Interaktion um ein sehr interdisziplinäres Feld: Um für den Menschen gut geeignete Schnittstellen schaffen zu können, ist Wissen um die Funktionsweise des menschlichen Geistes und Körpers notwendig. Für eine fundierte Bewertung eines System sind qualitative und quantitative Verfahren notwendig, die das empirische Überprüfen von Hypothesen ermöglichen. Die hier eingesetzten Verfahren stammen häufig aus der Psychologie und Soziologie. Für die Auswertung erfasster Daten sind statistische Kenntnisse

notwendig. Um einen Prototypen oder ein finales System umzusetzen sind die entsprechenden technischen Fähigkeiten gefragt – hier kommen Informatik, Physik und Ingenieurswissenschaften zum Einsatz.

2.1.1 Menschliche Faktoren

Die physiologischen und psychologischen Eigenschaften des Menschen, die bei der Nutzung eines Systems eine Rolle spielen, werden oft unter dem Stichwort „Menschliche Faktoren“ zusammengefasst. Am Beispiel der Texteingabe kann damit sowohl die erforderliche Feinmotorik zum Treffen der Tasten gemeint sein, als auch das Sehvermögen zum Erkennen der Tasten. Mental ist das Abstraktionsvermögen zum Umsetzen von Gedanken in einzelne Buchstaben notwendig, die durch Tastendrücke eingegeben werden um auf einem Bildschirm zu erscheinen. Es gibt zwar viele allgemeine Regeln, die beim Entwurf eines Systems für menschliche Nutzung berücksichtigt werden müssen, die hohe Diversität bei der Ausprägung der verschiedenen Merkmale erfordert jedoch oft ein Abwägen und das Eingehen von Kompromissen bei der Gestaltung eines Systems für menschliche Nutzung. Spezielle Anpassungen an eine bestimmte Zielgruppe gehen manchmal zu Lasten des „Durchschnittsmenschen“, ermöglichen Mitgliedern dieser Zielgruppe jedoch ein deutlich verbessertes Nutzungserlebnis.

Im Folgenden wird beschrieben, welche menschlichen Eigenschaften und Anforderungen bei der Erstellung und Beurteilung von Programmen speziell für Smartphones berücksichtigt werden müssen und in welchen Dimensionen eine Beurteilung vorgenommen werden kann.

2.1.2 Usability und User Experience

Neben den funktionalen Anforderungen an ein Programm, also die Erfüllung der Aufgaben, für die das Programm entwickelt wurde, gibt es gerade in Hinblick auf die Nutzer des Programms oft noch nicht-funktionale Anforderungen. Dies kann zum Beispiel die Geschwindigkeit des Programms sein, aber auch die Art und Weise, wie der Nutzer mit dem Programm interagiert und wie die Bedienlogik des Programmes aufgebaut ist.

Zur Beurteilung solcher Faktoren werden oft die Begriffe *Usability* (dt.: Gebrauchstauglichkeit) und *User Experience* (dt.: Nutzungserlebnis) verwendet. *Usability* beschreibt, wie effektiv und effizient sich ein Programm zur Erreichung eines Ziels verwenden lässt. Die International Organization for Standardization (ISO) definiert Usability so:

> Usability: extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use. (ISO 9241-210:2.13, International Organization for Standardization 2010)

Gebrauchstauglichkeit ist also ein Maß, mit dem beurteilt werden kann, wie gut ein System, Produkt oder Dienst von bestimmten Nutzern in einem spezifischen Kontext verwendet werden kann, um definierte Ziele effektiv, effizient und zufriedenstellend zu erreichen. Wichtig ist, dass eine Vergleichbarkeit immer nur in einem bestimmten Kontext gegeben ist. Wenn sich einer der angegebenen Faktoren ändert, muss die Gebrauchstauglichkeit neu beurteilt werden, ein Vergleich ist nicht mehr möglich.

Gebrauchstauglichkeit wird durch viele verschiedene Dimensionen beeinflusst: Neben in der Physiologie begründeten Aspekten gehört dazu auch, wie die Bedienlogik gestaltet ist, ob die Software die Bedürfnisse der Nutzer bezüglich Erlernbarkeit und Funktionsumfang erfüllt sowie die Befriedigung der Nutzer bei der Verwendung des Programms.

User Experience, oft auch nur kurz UX, ist weiter gefasst und beurteilt nicht nur, wie gut eine Aufgabe mit einem Programm lösen lässt, sondern auch wie der Nutzer die Verwendung des Programms antizipiert und daran zurück denkt. Wiederum nach der ISO:

> User Experience: person's perceptions and responses resulting from the use and/or anticipated use of a product, system or service (ISO 9241-210:2.15, ebd.)

Bei der Beurteilung der Nutzererfahrung wird mit Wahrnehmung und Reaktion die gesamte Bandbreite an Emotionen einer Person herangezogen, die mit der Nutzung des Programms oder Systems verbunden sind – direkt oder indirekt. Oft bewirkt eine höhere Usability auch eine bessere User Experience, da ein anstrengungsfreies Lösen einer Aufgabe von Nutzern als angenehmer empfunden wird. Ein sehr effizientes, oft wiederholtes Lösen einer Aufgabe mit einem Programm mit guter Usability kann jedoch trotzdem mit einer schlechte User Experience verbunden sein, wenn sich der Nutzer zum Beispiel langweilt und Abwechslung vermisst.

Die Berücksichtigung der Gebrauchstauglichkeit von Computersystemen (und anderen Arbeitsmitteln) ist sehr wichtig für den Einsatz im Alltag. In der Forschung hingegen und beim Ausprobieren neuer Ideen kann eine strikte Beurteilung einer Idee oder eines Prototypen anhand von Usability-Kriterien schädlich sein:

> If done during early stage design, it can mute creative ideas that do not conform to current interface norms. (Greenberg und Buxton 2008)

Das Spektrum an Tests und Untersuchungstechniken zur Beurteilung von Usability und User Experience ist sehr weit. Neben der persönlichen Erfahrung des Untersuchenden ist die Technik des *Human-centered design*, des Mensch-zentrierten Entwicklungsprozesses sehr wichtig. In diesem werden bei der Entwicklung eines Produktes die zukünftigen Nutzer des Produkts sehr früh mit einbezogen und in den verschiedenen Entwicklungsstufen jeweils dazu eingeladen, Feedback zum aktuellen Stand zu geben. Das eingesetzte Repertoire kann dabei je nach Entwicklungsstufe und der Art der benötigten Informationen von Interviews über Tests mit verschiedensten Prototypen bis hin zum standardisierten Beurteilungsbogen reichen.

Interviews

Interviews ermöglichen es, in einer eher ungezwungenen Atmosphäre mit einem Experten oder einem Mitglied der Zielgruppe eines Produkts zu sprechen. Das Spektrum reicht dabei von komplett freien Gesprächen bis hin zu strukturierten Interviews, die auf Basis einer schriftlichen Gesprächsvorlage erfolgen und es ermöglichen, mit verschiedenen Personen geführte Interviews systematisch vergleichbar zu gestalten.

Sie sind oft ein guter Weg, zu Beginn einer Untersuchung einen Überblick über relevante Themen zu erhalten, auf deren Basis dann zum Beispiel Anforderungen festgehalten und Fragebögen gestaltet werden können. Später ermöglichen sie es den Versuchspersonen, Anmerkungen anzubringen nach denen ansonsten nicht explizit gefragt wurde.

Schwierig kann sein, die in verschiedenen Interviews enthaltenen Informationen nach den gleichen Maßstäben zu kodieren, was eine statistische Auswertung erschwert. Zusätzlich kann es nötig sein, die Gespräche für die spätere Auswertung zu transkribieren, was den Nachbereitungsaufwand deutlich erhöht.

Fragebögen

Fragebögen dienen der strukturierten Erfassung von Informationen. Neben Freitextfragen sind meistens Fragen mit einer vorgegebenen Bewertungsskala enthalten. Diese Bewertungsskalen können je nach Typ nominal (z.B. Geschlecht), ordinal (z.B. „gut/-mittel/schlecht“), intervallbasiert (z.B. Temperaturen) oder rational (z.B. Zeiten) sein. Die so erfassten Daten lassen sich gut statistisch auswerten.

Fragebögen sind in der Regel effizienter zu nutzen als Interviews, da sie unbeaufsichtigt und von mehreren Personen gleichzeitig ausgefüllt werden können. Allerdings sind sie aufgrund der schriftlichen Fixierung der Frage eingeschränkter bezüglich der erfassbaren Informationen.

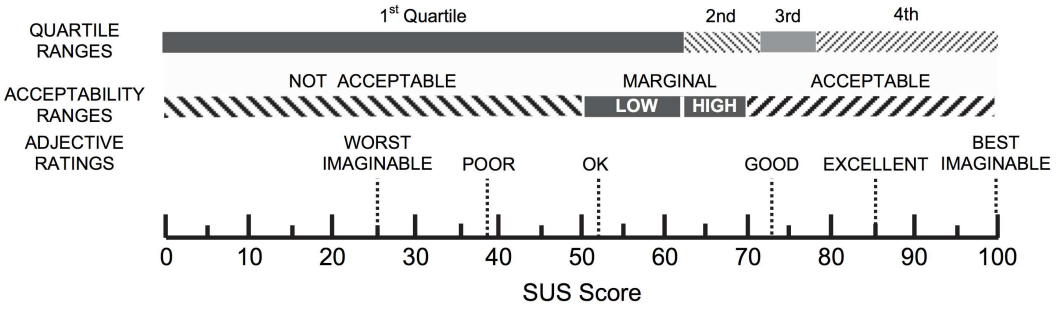


Abbildung 2.1: Ein Vergleich der SUS-Scores nach Quartilen, adjektivischer Beschreibung und Akzeptanz-Rating (Abb. aus Bangor, Kortum und Miller 2008)

System Usability Scale Ein häufig verwendeter Fragebogen für die schnelle Einschätzung der Usability eines Systems ist der von Brooke (1996) entwickelte *System Usability Scale*, oft mit *SUS* abgekürzt. Er besteht aus zehn Fragen, die auf einer nach Likert (1932) benannten Skala mit fünf Abstufungen beantwortet werden müssen. Zur Auswertung des Fragebogens wird ein sog. *SUS-Score* berechnet, der von 0 bis 100 reichen kann, wobei ein größerer Werte für eine bessere Usability steht. Der Score wird aus der Gesamtheit der Fragen berechnet. Brooke warnt explizit davor, Items des Fragebogens einzeln zu interpretieren. Details zur Berechnung des Scores finden sich in der Originalveröffentlichung.

Zehn Jahre nach Veröffentlichung des *SUS* untersuchten Bangor et al. die empirische Anwendung des *SUS* anhand einer Vielzahl darauf basierender, publizierter Studien (Bangor, Kortum und Miller 2008). Diese Untersuchung bestätigt, dass der SUS-Score nur im Gesamten verwendet werden darf. Außerdem untersuchten sie, wie sich die erzielten Scores mit Worten beschreiben lassen. Ein Ergebnis über 50 bedeutet dabei, dass das untersuchte Systeme eine ausreichende (org.: *OK*) Gebrauchstauglichkeit aufweist. Werte über 70 sind akzeptabel, ab 73 kann man von gut sprechen, über 85 ist der Wert exzellent (vgl. Abb. 2.1).

Teilweise muss der Fragebogen allerdings an seinen jeweiligen Einsatzzweck angepasst werden. So wurde das Wort „cumbersome“ in Item 8 von mehreren Testpersonen nicht verstanden und mit „awkward“ ersetzt (Bangor, Kortum und Miller 2008; Finstad 2006). Zudem wird von Bangor, Kortum und Miller (2008) basierend auf Rückmeldung von Testpersonen vorgeschlagen, im Fragebogen das Wort „system“ durch „product“ zu ersetzen. Für einen Einsatz mit Testpersonen, die kein Englisch sprechen, muss der Fragebogen generell übersetzt werden. Eine offizielle deutsche Übersetzung existiert jedoch nicht. Nach einer Recherche wurde für diese Arbeit die Version von Lohmann (2013) ausgewählt.

Fitts' Gesetz

Einer für die Gestaltung von Nutzeroberflächen sehr bedeutsamen Frage widmete sich der Psychologe Paul Fitts (1954) in seiner Veröffentlichung „The information capacity of the human motor system in controlling the amplitude of movement.“: Was ist wichtiger für die Erledigung feinmotorisch anspruchsvoller Zeigeaufgaben: Das Gehirn oder der Muskelapparat?

Der von ihm entdeckte Zusammenhang findet sich sehr vielen Aspekten der Usability-Forschung. Er beschreibt in Anlehnung an die Informationstheorie nach Shannon (2001), wie viel kognitive Ressourcen im motorischen System des Menschen als Kommunikationskanal für bestimmte Steuerungs- oder Zeigeaufgaben notwendig sind. Dabei hängt die zur Lösung der Aufgabe benötigte Zeit linear von den Fähigkeiten des jeweiligen Nutzers bei der Nutzung des untersuchten Systems sowie einem sogenannten „Schwierigkeitsindex“ (engl. *Index of difficulty*, I_d) ab, der eine inhärente Eigenschaft der jeweiligen konkreten Aufgabenstellung ist. Generell gilt: Je kleiner und weiter entfernt das Ziel einer Zeigeaufgabe ist, desto schwieriger wird die Aufgabe und desto länger benötigt ein Mensch für ihre fehlerfreie Durchführung.

Der „Index of Difficulty“ nach Fitts Der „Index of Difficulty“ beschreibt, wie viel Bandbreite in Bit im motorischen System als Kommunikationskanal für bestimmte Steuerungsaufgaben notwendig ist:

$$I_d = -\log_2 \frac{W_s}{2A} \text{ Bit} \tag{2.1}$$

Dabei bezeichnet W_s die Breite des Ziels und A die Amplitude der Bewegung, mit der das Ziel erreicht werden kann. Das Verhältnis von W_s zu A bestimmt I_d: Je kleiner das Ziel im Verhältnis zur zurückzulegenden Entfernung ist, desto schwieriger wird die Aufgabe. Eine anschauliche Interpretation könnte lauten: Je größer die Entfernung zwischen zwei Zielen ist, desto mehr beschleunigt ein Nutzer Hand und Arm, um die Distanz zurückzulegen. Um ein sehr kleines Ziel zu treffen, muss sie oder er den Arm erst abbremsen. Dann kann man sie/er sich auf den neuen Kontext konzentrieren, das Ziel erfassen und es mit dem Finger berühren. Wenn die Entfernung zwischen zwei Zielen jedoch nur klein ist, entfallen diese „Brems-Phase“ und der mentale Kontextwechsel und es kann sofort mit der feinmotorischen gesteuerten Berührung des Ziels begonnen werden.

Finale Formulierung von Fitts' Gesetz Aus dem „Index of Difficulty“ wurde von verschiedenen Autoren der Zusammenhang abgeleitet, der gemeinhin als „Fitts' Gesetz“ bekannt ist und die Zeit t beschreibt, die für das Berühren eines Ziels notwendig ist.

Aus verschiedenen Gründen muss die Formel noch erweitert werden: Wenn der Nutzer durch ein Signal zum Starten des Experiments aufgefordert wird, so muss die Reaktionszeit a zwischen dem Signal und dem tatsächlichen Starten der Ausführung berücksichtigt werden. Außerdem wird die Schwierigkeit einer Aufgabe je nach Testperson unterschiedlich wahrgenommen. Um diese persönliche Leistungsfähigkeit auszudrücken, wird der „Index of Difficulty“ normalerweise mit einem Vorfaktor b versehen. Außerdem hat sich die Invertierung des Verhältnis von Durchmesser und Distanz etabliert, um die Subtraktion des I_d durch eine Addition zu ersetzen.

Nachdem diese Anpassungen nicht alle zeitgleich vorgeschlagen wurden, existieren in der Literatur verschiedene Formulierungen der sich ergebenden Formel. Ohne genaue Kenntnis der verwendeten Formel wird dadurch das Vergleichen von Ergebnissen erschwert, die auf Fitts' Gesetz aufbauen. Drewes (2010) schlägt daher die allgemeine Verwendung der folgenden, auf Fitts' Formulierung des I_d aufbauenden Formel vor:

$$t = a + b \cdot \log_2 \frac{2A}{W_s} \qquad \text{Fitts' Law}$$

Generell und unabhängig von der genauen Formulierung der Formel kann man jedoch ableiten, dass große, nahe Ziele schneller erfolgreich berührt werden können als weiter entfernte Ziele geringer Größe. Wo sehr kleine Ziele berührt werden müssen, wird der Nutzer gezwungen entweder mit einer hohen Fehlerrate zu leben oder die Aufgabe konzentriert und entsprechend langsam vorzunehmen.

2.1.3 Besonderheiten mobiler Nutzungsszenarien

Im Vergleich zu stationären Computersystemen stellt eine mobile Nutzung spezielle Anforderungen an Hard- und Software. Dies gilt insbesondere für Smartphones, da diese in grundlegend anderen Kontexten verwendet werden als zum Beispiel ein Desktop-PC. Die wichtigsten Unterschiede sind:

- Sehr diverse Nutzungskontexte, da ein Smartphone ein ständiger Begleiter in allen Lebenslagen ist.
- Fragmentierte Aufmerksamkeit, da eine Nutzung oft parallel zu anderen Aufgaben stattfindet.
- Variable Interaktionsmodalitäten, in Abhängigkeit von der Nutzungssituation.
- Eingeschränkte Ressourcen, da das Gerät unterwegs nicht beliebig Strom oder Bandbreite zum Internet nutzen kann.
- Einschränkungen der Privatsphäre, da eine Nutzung auch in Gesellschaft anderer Menschen erfolgt.

Was dies bedeutet, wird im Folgenden ausführlicher beschrieben:

Diverse Nutzungskontexte

Mobile Geräte dienen heute als nahezu ständiger Begleiter und decken einen Vielzahl an Funktionen ab. Durch ihre ubiquitäre Verwendung variieren auch die Umstände der Nutzung stark, vor allem im Gegensatz zu einem klassischen Desktop-Computer. Während dieser in der Regel sitzend in einem Raum mit kontrollierten Umweltbedingungen bedient wird, begleiten Smartphones ihren Nutzer morgens vom Wachwerden im Bett über den Weg zur Arbeit bis hin zum abendlichen Fernsehen auf dem Sofa (und durch alle anderen denkbaren Situationen im Alltag). Diese Vielfalt macht es zum einen schwer, für eine Interaktionsaufgabe die eine perfekte Lösung zu finden, zum anderen kann beim Entwickeln von Anwendungen in der Regel kein bestimmter Nutzungskontext vorausgesetzt werden.

Fragmentierte Aufmerksamkeit

Ebenfalls bedingt durch die fortwährende Nutzung, die oft auch nicht beim Fortbewegen im öffentlichen Raum stoppt, kann die Aufmerksamkeitsspanne des Nutzers sehr kurz sein. Oulasvirta u. a. (2005) untersuchten, wie sich beim Nutzen von Webseiten und gleichzeitigem Navigieren in der Öffentlichkeit die Aufmerksamkeit der Nutzer auf Gerät bzw. Umwelt richtete. Aufmerksamkeitsspannen zwischen vier und acht Sekunden waren dabei typisch, bei längeren Wartezeiten wechselte der Fokus mehrere Male zwischen Gerät und Umgebung hin und her.

Variable Interaktionsmodalitäten

Ein weiterer variabler Faktor ist die „Handhabung“ der Nutzer. Noch sind Smartphones so gestaltet, dass sie einhändig gehalten werden können[1]. Die Bedienungs des Geräts erfolgt meist über den Bildschirm, der berührungsempfindlich ist und je nach Gerät auch für Hover Detection verwendet werden kann. Für die Bedienung können eine oder zwei Hände verwendet werden. Bei der beidhändigen Variante gibt es Nutzer, die das Gerät mit einer Hand halten und mit den Fingern der anderen Hand oder einem von dieser gehaltenen Stylus bedienen. Alternativ kann das Gerät auch mit beiden Händen gegriffen und mit den Daumen bedient werden; diese Haltung findet vor allem bei der Verwendung der Bildschirmtastatur Anwendung. Diese Art der Daumen-Nutzung funktioniert auch mit nur einer Hand, allerdings kann die eingeschränkte Beweglichkeit des Daumens ein Problem darstellen (Park und Han 2010), vor allem wenn das Gerät dem aktuellen Trend zu großen Bildschirmdiagonalen folgt (Löchtefeld u. a.

[1] Ausnahmen sind die sog. „Phablets“ (von engl. *phone* und *tablet*), die zwar die Funktionen eines Mobiltelefons haben, sich aber eher an der Größe eines Tablets orientieren.

2015). Zwischen der ein- und beidhändigen Nutzung kann je nach Bedarf spontan gewechselt werden. In der Regel wird dabei die jeweils dominante Hand verwendet (Arif 2012).

Nutzung eines berührungsempfindlichen Bildschirm Obwohl der Bildschirm eines modernen Smartphones eine vergleichbare Auflösung hat wie ein üblicher PC-Monitor (1920×1080 Bildpunkte oder höher), ist er absolut gesehen deutlich kleiner. Gleichzeitig ist durch die Bedienung mit dem Finger die Genauigkeit im Vergleich zur Nutzung einer Maus herabgesetzt: Bei der Berührung des Bildschirms kann nicht genau vorher gesagt werden, an welcher Stelle der erste Kontakt mit dem Bildschirm und damit die vom System festgestellte Position sein wird. Umgangssprachlich wird dies auch als „Fat Finger Syndrom“ bezeichnet. Für eine ergonomische Nutzbarkeit muss eine Mindestgröße der interaktiven Elemente eingehalten werden (vgl. 2.1.3), insgesamt ist deshalb weniger Platz für gleichzeitig dargestellte Steuerelemente als auf einem PC-Bildschirm.

Die Bedienung mit dem Finger hat noch einen weiteren Nachteil: Der Nutzer verdeckt das Ziel seiner Interaktion und weitere Teile des Bildschirms. Diesem Effekt kann durch eine versetzte Anzeige entgegen gewirkt werden. Vogel und Baudisch (2007). fanden heraus, dass eine versetzte Anzeige des Bereichs unter dem Finger auf einem PDA die sichere Bedienung von Software erlaubt, die ursprünglich auf die Nutzung mit einem (dünnen und sehr genauen) Stylus ausgelegt wurde, vor allem wenn die versetzte Anzeige gleichzeitig auch vergrößernd wirkte (vgl. Abb. 2.2). Dies wurde von Apple für die Bildschirmtastatur des iPhone übernommen. Bei der Berührung eines Buchstaben erscheint eine kleine, versetzte Anzeige zur Bestätigung des Ziels. Mit diesem Feedback kann der Nutzer bei Berührung des falschen Ziels seine Auswahl durch Verschieben des Fingers korrigieren und den richtigen Buchstaben erst durch Heben des Finger auslösen (Martin u. a. 2009). Inzwischen bieten alle aktuellen Smartphone-Betriebssysteme eine vergleichbare Funktionalität an.

Die empfohlenen Zielgrößen interaktiver Elementen variieren in Abhängigkeit der Größe des verwendeten Fingers. Bei Bedienung mit dem (breiten) Daumen liegen sie zwischen 8 mm und 10 mm (Parhi, Karlson und Bederson 2006; Park und Han 2010). Wird das Gerät mit einer Hand festgehalten und mit dem Zeigefinger der anderen Hand bedient, so sinken die empfohlenen Mindestgrößen auf 7 mm bis 9 mm (Microsoft Inc. 2016).

Beim Gehen mit gleichzeitiger Eingabe von Text hängt die Texteingabeleistung von der Komplexität der zu bewältigenden Wegstrecke ab. Komplexere Pfade vermindern dabei wie von Fitts' Gesetz vorhergesagt entweder die Geh-Geschwindigkeit oder die Genauigkeit beim Berühren eines Ziels, da die mentale Bandbreite nicht zum gleichzeitigen Lösen beider Aufgaben ausreicht (Lin u. a. 2007).

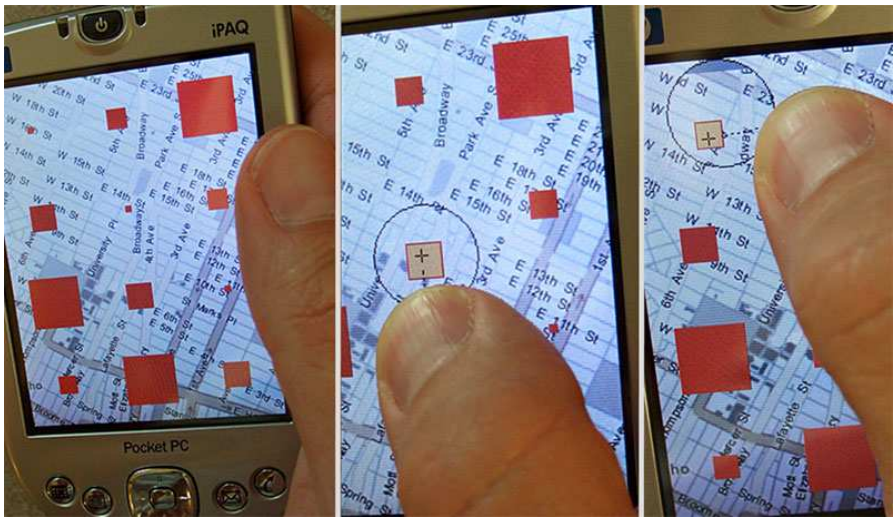


Abbildung 2.2: Versetzte Zieldarstellung mit „Shift“ (Abb. aus Vogel und Baudisch 2007)

Umgebungssensoren Der Reiz moderner Smartphones liegt nicht zuletzt auch in den vielfältigen Sensordaten, die sie zur Verfügung stellen. Die damit erfassten Daten können genutzt werden, um dem Nutzer bessere Anwendungen anzubieten (zum Beispiel die aktuellen Position und Bewegungsrichtung über GPS und Magnetometer). Ebenso ist eine Anpassung des Geräts an den aktuellen Nutzungskontext (Lagesensor für angepasste Bildschirmausrichtung, Mikrofon zur Unterdrückung von Störgeräuschen, Helligkeitssensor zur Anpassung der Bildschirmhelligkeit) möglich.

Eingeschränkte Ressourcen

Problematisch bei der Verwendung von Mobilgeräten ist weiterhin, dass Stromversorgung und Netzwerkanbindung meist limitiert sind. Aufgrund der zahlreichen übernommenen Funktionen und der Relevanz für den Alltag ist es den meisten Nutzern jedoch sehr wichtig, dass das Gerät den ganzen Tag über zur Verfügung steht. Auch wenn sich die Hardware-Hersteller bemühen, die Geräte sparsamer zu machen und gleichzeitig größere Akkus zu verbauen, sollten App-Entwickler Rücksicht nehmen und den Ressourcenverbrauch ihrer Anwendungen entsprechend planen. Nutzer können zusätzliche Maßnahmen ergreifen, damit der Akku nicht vorzeitig geleert wird, zum Beispiel die Rechenleistung des verbauten Prozessors oder die Helligkeit des Bildschirms verringern. Die Verwendung hoher Kontraste innerhalb einer Anwendungen kann helfen, sie unter solchen Umständen trotzdem noch gut verwendbar zu gestalten.

Auch die Anbindung an das Internet ist in der Regel limitiert. In Deutschland dominierten während der letzten Jahre Mobilfunkverträge, bei denen nur eine bestimmte

Datenmenge mit hoher Geschwindigkeit übertragbar ist. Danach wird die Bandbreite so weit eingeschränkt, dass nur noch textbasierte Übertragungen sinnvoll funktionieren.

Privatsphäre im öffentlichen Raum

Da Smartphones in so gut wie allen Lebensbereichen verwendet werden, verarbeiten sie auch Informationen privater Natur. Gerade in hoch-mobilen Szenarien wie einer Nutzung im öffentlichen Nahverkehr sollte dabei trotzdem die Privatsphäre der Nutzer gewahrt bleiben. Hilfsmittel wie Sprachsynthese oder Spracherkennung sind daher nicht uneingeschränkt nutzbar, obwohl sich deren Qualität und Zuverlässigkeit in den letzten Jahren deutlich verbessert hat.

2.1.4 Smartphone-Nutzung im Alter

Besteht die Zielgruppe für Nutzung eines Computersystems aus Senioren, so verstärken sich einige der bisher angesprochenen Problematiken. Mit zunehmendem Alter treten verschiedene psychologische und physiologische Alterserscheinungen auf, welche die Nutzung von Smartphones beeinflussen können. Generell wird im Alter das Erlernen von automatisierten Prozessen schwieriger. Daher ist es wichtig, dass auf bestehenden Kenntnissen aufgebaut werden kann, da diese im Alter erhalten bleiben (Fisk u. a. 2009).

Konzeptionell kommt die Bedienung eines interaktiven Systems über einen Touchscreen älteren Nutzern entgegen, da im Alter die Leistung des Arbeitsgedächtnisses nachlassen kann (Kester u. a. 2002). Touchscreens stellen ein besonders geeignetes Interaktionsmedium dar, da sie ein direktes Berühren eines Ziels erlauben und ihre Nutzung nur eine geringe mentale Belastung verursacht. Diese Aspekte wurden bereits früh von Shneiderman (1991) postuliert, aber erst in neuerer Zeit empirisch untersucht (McLaughlin, Rogers und Fisk 2009). Im Gegensatz zur Nutzung einer Maus (ein indirektes Zeigegerät) war die Leistung in Zeigeaufgaben sowohl bei Verwendung des Fingers (Murata und Iwase 2005) als auch eines Stylus (Siek, Rogers und Connelly 2005) bei Jugendlichen, Menschen mittleren Alters und älteren Personen vergleichbar.

Nachlassende feinmotorische Kontrolle der Hand erschwert das zielgerichtete Betätigen von Schaltflächen (Carmeli, Patish und Coleman 2003), auch wenn dies teilweise durch Vorausplanen der Bewegung und eine niedrige Geschwindigkeit ausgeglichen werden kann (Walker, Philbin und Fisk 1997). Die Interaktion mit einem Touchscreen kann durch trockene Haut an den Fingern, ein häufig auftretendes Altersphänomen, zudem rein physikalisch erschwert werden (White-Chu und Reddy 2011).

Ein weiteres Problem kann nachlassende Sehkraft sein. Altersweitsichtigkeit tritt auf, wenn das Auge aufgrund nachlassender Elastizität der Linse nicht mehr bis in den Nahbereich scharf stellen kann (Schieber 2003). Mitte des vierten Lebensjahrzehnts beginnen die Schwierigkeiten, auf Armlänge normal gedruckten Text lesen zu können (Atchison, Capper und McCabe 1994). Text auf Smartphones, der oft noch kleiner dargestellt wird, ist dann natürlich ebenfalls nicht mehr gut lesbar. Obwohl diese Weitsichtigkeit mit Hilfe einer Brille einfach korrigiert werden kann, werden diese Lesebrillen – häufig wegen negativer Affekte auf die Selbstwahrnehmung oder aus praktischen Gründen – nicht verwendet. Mit zunehmendem Alter treten allerdings auch andere Sehschwächen auf, die hauptsächlich auf Veränderungen der Retina basieren und nicht mehr einfach mit einer Brille korrigiert werden können (Schieber 2003).

Wie müssen Hard- und Software angepasst werden, um diese Effekte auszugleichen? Wo möglich, ist eine Vergrößerung der Interaktionselemente zum Ausgleichen dieser Alterserscheinungen vielversprechend (Jin, Plocher und Kiff 2007). Aufgrund des relativ kleinen Formfaktors von Smartphones ist dies jedoch nicht unbegrenzt möglich. Immerhin bieten inzwischen die meisten mobilen Betriebssysteme an, die Größe der Schriftdarstellung anzupassen und so mehr auf die Bedürfnisse von Nutzern mit eingeschränkter Sicht einzugehen (Klaus u. a. 2012). Allerdings kann die größere Darstellung die Übersicht der Nutzer über die dargestellten Informationen reduzieren – dies stellt also einen Kompromiss dar (Findlater und McGrenere 2008). Auch der Kontrast der Darstellung und die Lautstärke sind meist anpassbar und tragen damit dazu bei, ein handelsübliches Smartphone verwenden zu können, anstelle eines speziellen Senioren-Gerätes (Kurniawan 2008), was mit einer empfundenen Stigmatisierung einhergehen könnte (Klaus u. a. 2012). Einige Probleme ließen sich schon durch Anpassung der Richtlinien für die Gestaltung der Touchscreen-Oberflächen beheben, die momentan eher auf jüngere Altersschichten zugeschnitten sind (Jin, Plocher und Kiff 2007).

2.2 Texteingabe auf mobilen Geräten

Der am häufigsten verwendete Mechanismus, um an Computern Text einzugeben, ist die Tastatur. Die Anordnung der Tasten ist soweit standardisiert, dass viele Menschen inzwischen ohne hinsehen zu müssen ihre über 100 Tasten blind bedienen können. Was man einer Tastatur dabei nicht ansieht: Ihre Entwicklung ging der des Computers deutlich voraus.

2.2.1 Die Entstehung des QWERTY-Layouts

Die prinzipielle Funktionsweise und die Anordnung der Tasten erinnern immer noch deutlich an die Tastaturen, die zu Beginn des 19. Jahrhunderts mit der Einführung von Schreibmaschinen aufkamen. Damals wurden die Tasten verwendet, um eine komplexe

Mechanik in Gang zu setzen, mit der schließlich der Buchstabe durch einen Hebel auf das Papier gedruckt wurde. Diese Mechanik war fehleranfällig, wenn nebeneinander liegende Tasten kurz nacheinander oder gleichzeitig betätigt wurden, die Hebel konnten sich beim Schlag auf das Papier berühren und verklemmten in Folge. Daher entwickelte Christopher Sholes gegen Ende des 19. Jahrhunderts das heute noch verwendete QWERTY-Layout, bei dem er darauf achtete, häufig vorkommende Buchstabenpaare räumlich zu trennen. Ergonomie oder Nutzerfreundlichkeit waren noch nicht das Ziel (David 1985).

Es überrascht nicht, dass sich bereits eine Vielzahl von Wissenschaftlern mit der Verbesserung dieser Eingabeform auseinandergesetzt haben, um einige der tatsächlichen oder empfundenen Probleme dieser vermeintlich suboptimalen Lösung zu beseitigen. Nur wenige dieser Verbesserungen konnten sich jedoch durchsetzen. Obwohl national angepasste Tastaturlayouts existieren, bei denen einzelne Tasten vertauscht wurden um den lokalen Sprachen Rechnung zu tragen[2], konnte die generelle QWERTY-Anordnung für lateinische Schriftzeichen bis heute nicht verdrängt werden. Selbst vermeintlich kleine Änderungen wie das in den 30er Jahren von August Dvorak entwickelte „Dvorak Simplified Keyboard“ mit seinen effizienter angeordneten Tasten (Lewis, Potosnak und Magyar 1997) oder ergonomischer aufgebaute Tastaturen (Hobday 1985) wurden von den Nutzern nicht angenommen – offensichtlich ist die rechteckige, flache QWERTY-Tastatur zumindest so gut, dass ein daran gewöhnter Nutzer nicht die Schwierigkeiten eines Umtrainierens auf sich nehmen will (Norman und Fisher 1982).

Nach neuerer Forschung ist die Anordnung der Tasten allerdings auch kein wesentlicher Faktor bei der Nutzung einer Tastatur (ebd.). Geübte Nutzer erreichen auch mit QWERTY-Layout über 400 Anschläge pro Minute; der Weltrekord[3] liegt im Moment bei 821 Anschlägen pro Minute.

Sehr deutliche Unterschiede bestehen bei der alternativen Belegungen der Tasten: Sonderzeichen sind durch das gleichzeitige Drücken von einer oder mehreren Modifikatoren-Tasten erreichbar und unterscheiden sich je nach verwendetem Betriebssystem teilweise deutlich, wie in Abb. 2.3 aufgeführt.

2.2.2 Kurze Geschichte mobiler Texteingabe

Gerade mobile Geräte sind in der Regel kleiner (und leichter) als stationäre Geräte. Je geringer die Abmessungen dabei werden, desto weniger Platz steht natürlich für eine Tastatur zur Verfügung. Je nach Geräteklasse wurden dabei folgende Kompromisse eingegangen:

[2] Zum Beispiel das französische AZERTY– oder das deutsche QWERTZ-Layout

[3] Helena Matouskova, 2001 in Hannover: 30-Minuten-Schnellschreiben, 24630 Anschläge, vier Fehler (Informationsverarbeitung 2001).

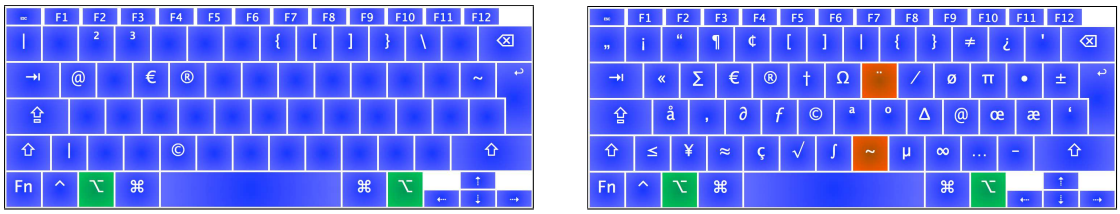


Abbildung 2.3: Unterschiedliche Sonderzeichen-Belegung im Tastatur-Layout von Windows (links) und Mac (rechts).

Tastaturen in mobilen PCs

In Notebooks wird normalerweise zumindest der Buchstabenblock einer normalen QWERTY-Tastatur eingebaut. Die Größe der Tastatur von der Größe des verbauten Bildschirms ab, der die Abmessungen des Gehäuses vorgibt. Sollte dieses zu klein werden, wie zum Beispiel bei Geräten mit 12 ″ Bildschirmdiagonale, dann wird normalerweise die Größe der einzelnen Tasten so weit reduziert, dass die Tastatur komplett verbaut werden kann[4]. Weitere Anpassungen werden oft an der Mechanik der Tasten vorgenommen, um Gewicht und Größe einzusparen. Mechanische Taster werden so etwa durch Scheren-Mechanismen oder auch gedruckte Schaltkreise ersetzt, bei denen bei Tastenbetätigung durch eine Plastikmatte ein Kontakt hergestellt wird. Meist lässt sich die Tastatur bei diesen Geräten ohne längere Gewöhnungsphase nutzen.

Tastaturen in „Taschencomputern“

Anders sieht es bei Geräten aus, die dazu gedacht sind in Hosen– oder Jackentaschen transportiert zu werden.

Erste Mobiltelefone In Anlehnung an ihre Telefonfunktion waren die ersten Mobiltelefone mit einem numerischen 12-Tasten-Feld ausgestattet, wie es von stationären Apparaten her bekannt war. Für die Texteingabe sind die Tasten in alphabetischer Reihenfolge mehrfach belegt, so sind beispielsweise die Buchstaben „A“, „B“ und „C“ auf der Taste „2“ zu finden (vgl. Abb. 2.4a). Im Texteingabemodus erfolgt die Eingabe des „A“ durch einen einfachen Tastendruck, die des „B“ durch einen doppelten Tastendruck. Durch einen Timeout oder Betätigen einer speziellen Tasten wird die Eingabe eines Buchstabens beendet und die Eingabemarke weiter gerückt.

[4] Eine Ausnahme stellt das IBM ThinkPad 701 von 1995 dar, bei dem das „Butterfly Keyboard“ in zwei Teile getrennt Platz im Gehäuse hat. Beim Öffnen schieben sich die beiden Hälften der Tastatur nebeneinander und ragen über das Gehäuse des Notebooks heraus.

(a) Tastatur eines Nokia 6300 mit Buchstaben und Zahlen (basierend auf Atanasov 2008)

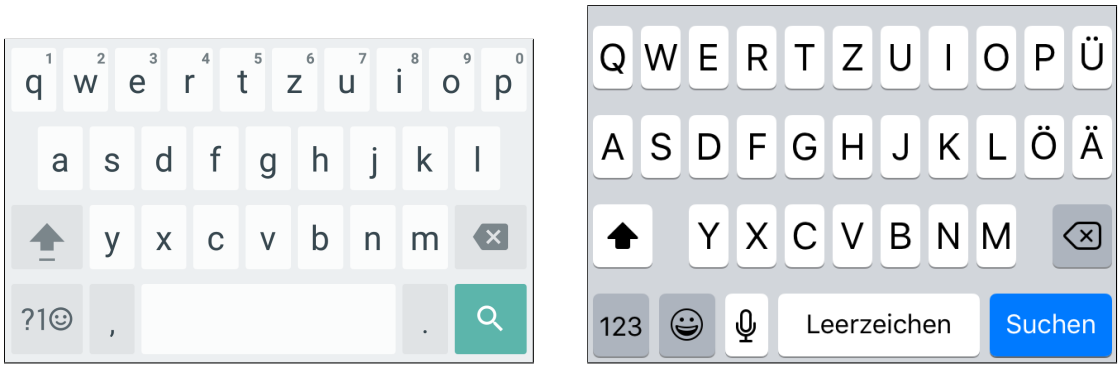
(b) Übersicht des Graffiti-Alphabets auf einem Palm PDA (Quelle: Palm OS)

Mit der Eingabetechnik „T9“ (engl. für „Text on 9 keys“) der Firma Nuance[5] wurde diese Art der Texteingabe effizienter gestaltet: Für die Eingabe eines Wortes muss die entsprechende Taste je Buchstabe nur noch einmal betätigt werden. Anhand der statistischen Analyse der Worthäufigkeiten in der Eingabesprache und der zuvor betätigten Tasten kann damit ein Wort vorgeschlagen werden, welches der Reihenfolge der betätigten Tasten entspricht. Für das Wort „smart“ müssen damit die Tasten `7-6-2-7-8` betätigt werden. Je nach verwendetem Wörterbuch kann diese Kombination jedoch auch ein anderes Wort ergeben, zum Beispiel „roast“. Zum Auflösen dieser Doppeldeutigkeiten gab es eine Zusatztaste, mit der durch die verschiedenen Treffer gewechselt werden konnte. Durch fortlaufende Anpassung der Wortvorschläge an die tatsächliche Nutzung und das Erlernen neuer Worte kann diese Texteingabetechnik noch weiter verbessert werden (Nuance 2017).

„Personal Digital Assistants“ Die anfangs noch nicht mit den Mobiltelefonen vereinte Geräteklasse der digitalen Assistenten, („Personal Digital Assistants“, PDA) bot mehr Platz, sodass sowohl Geräte mit einer physikalischen Tastatur als auch mit neuen Eingabetechniken auf den Markt gebracht wurden.

Geräte mit „echter“ Tastatur werden bis zu einer Größe von wenigen Zentimetern hergestellt, zum Beispiel vom Smartphone-Anbieter Blackberry. Wenn bei solch geringen Abmessungen allerdings gleichzeitig noch ein möglichst großer Bildschirm gewünscht wird, dann wird oft auf eine mechanische Tastatur verzichtet und statt dessen auf eine Alternative gesetzt, den berührungsempfindlichen Bildschirm (engl: „Touchscreen“).

[5] ursprünglich Tegic Communications



(a) Bildschirmtastatur unter *Android* (b) Bildschirmtastatur unter *iOS*

Abbildung 2.5: Beispiele für Bildschirmtastaturen gängiger Smartphones-Betriebssysteme

Bereits 1993 stelle Apple mit dem „Newton“ ein Gerät vor, das handschriftliche Texteingabe mittels eines speziellen Stiftes auf dem Bildschirm bot. Es konnte sich am Markt aber nicht durchsetzen. Ein erfolgreicheres Beispiel ist die softwarebasierte Eingabetechnik „Graffiti“, die auf Geräten mit *Palm OS* eingesetzt wurde. Mit eines Stylus wurden Gesten gezeichnet, die an die Form des jeweiligen Buchstabens im lateinischen Alphabet angelehnt sind (vgl. Abb. 2.4b). Geübte Nutzer waren mit dieser Eingabeart schneller als mit der ebenfalls verfügbaren auf dem Bildschirm dargestellten Tastatur (Fleetwood u. a. 2002). Die Darstellung der Buchstaben auf der Tastatur war allerdings so klein, dass sie nur mit dem Stylus gezielt berührt werden konnten.

Smartphones Inzwischen haben sich Smartphones durchgesetzt, die fast immer ohne mechanische Tastatur auskommen und bei Bedarf eine Bildschirmtastatur einblenden. Die mitgelieferten Tastaturen funktionieren alle sehr ähnlich. Typischerweise besitzen sie Tasten für alle Buchstaben des Alphabets, teilweise auch für Zahlen und Umlaute. Doppelt beschriftete Tasten weisen darauf hin, dass durch längere Berührung der Taste ein alternatives Zeichen ausgegeben werden kann. Ebenso sind Tasten für Zeilenvorschub, Leerzeichen und Löschen des letzten eingegebenen Zeichens vorhanden. Sonderzeichen können eingegeben werden, indem durch Betätigung einer speziellen Taste die angezeigten Tasten umgeschaltet werden. Großschreibung wird erreicht, indem vor dem gewünschten Buchstaben eine Umschalttaste berührt wird, woraufhin die Tastatur Großbuchstaben anbietet. Zusätzliche Funktion wie etwa Wortvorschläge sind populäre Erweiterungen dieses Basisumfangs.

Da Darstellung und Funktionsweise einer Bildschirmtastatur nur durch die entsprechende Software bestimmt werden, erlauben das Smartphone-Betriebssystem *Android* und inzwischen auch das von Apple eingesetzte *iOS* das Austauschen der Systemtastatur durch den Anwender. Dadurch konnten und können nun auf Touchscreens ba-

sierende alternative Eingabetechniken einfach im Massenmarkt getestet werden. Die einfache Installation über einen der App-Stores und der geringe Preis (wenige Euro bzw. Dollar) führen allerdings auch dazu, dass solche Eingabetechniken nun oft nicht mehr wissenschaftlich evaluiert, sondern direkt veröffentlicht werden und daher zu einigen Eingabetechniken keine wissenschaftlichen Publikationen mehr existieren.

Generell werden allerdings sowohl die etablierten als auch neu hinzukommende Texteingabelösungen nach den gleichen Kriterien bewertet. Da ein Großteil der individuellen Kommunikation inzwischen textbasiert über Mobiltelefone abgewickelt wird, sind neben den klassischen Faktoren der Nutzerfreundlichkeit heutzutage vor allem die Schnelligkeit und Genauigkeit bei der Texteingabe wichtig.

Spracheingabe als Alternative? Eine mögliche Variante zur Texteingabe wäre Spracheingabe, die allerdings noch immer „kurz vor dem Durchbruch“ steht und sich bis heute nicht als uneingeschränkt praxistauglich erwiesen hat: Wo ein Computersystem bei der Verwendung von Tastatur- oder grafischer Schnittstelle noch Hilfestellungen und Grenzen bei der Nutzung vorgeben kann, ist bei Spracheingabe keinerlei solche Kontrolle mehr möglich. Für eine wirklich sinnvolle Nutzung von Spracheingabe zur Steuerung eines Computers wäre es daher notwendig, dass dieser auch den Sinn und nicht nur den Text des Gesprochenen verstehen kann. Sprachgesteuerte Interaktion funktionierte lange nur mit spezifischen Wortschatz für begrenzte Aufgaben, den der Nutzer lernen musste. Heute ermöglichen Fortschritte im Feld der künstlichen Intelligenz sowie bei der Verarbeitung sehr großer Datenmengen eine semantische Interpretation von natürlichsprachlich eingegebenen Daten. Als Beispiele seien hier zwei Dienste genannt: Auf der Webseite von *Wolfram Alpha* kann der Nutzer eine Frage eingeben, die der Dienst versucht auf Basis einer semantischen Datenbank zu verstehen und zu beantworten. Der Schwerpunkt liegt dabei ganz klar auf zahlenbasiertem Wissen. Auf der anderen Seite steht die Assistenzfunktion *Siri* der Firma Apple Inc.: Hier kann der Nutzer per Spracheingabe sowohl allgemeine Suchanfragen stellen als auch die Grundfunktionen seines Smartphones bedienen und so zum Beispiel einen Termineintrag anlegen. Ein wirkliches Gespräch ist zur Zeit noch nicht möglich.

Doch nicht nur das semantische Verstehen stellt ein Problem dar, auch die rein akustische Komponente ist nicht einfach zu lösen: Auf der einen Seite ist nicht jede Umgebung leise genug, um zuverlässige Spracheingabe zu erlauben. Auf der anderen Seite will der Nutzer oft nicht, dass zufällig anwesende andere Personen mithören können, was er dem Computer zu sagen hat. Private Nachrichten, aufgerufene Webseiten oder einfach nur Passwörter sind Daten, die in der Regel nicht öffentlich bekannt sein sollen. Daher kann Spracheingabe keinen vollständigen Ersatz für die Texteingabe mit einer Tastatur darstellen.

2.2.3 Entwurf von Experimenten zur Messung der Texteingabeleistung

Die Forschungsmethoden im Bereich der Mensch-Maschine-Interaktion lehnen sich stark an die der anderen mit Menschen befassten empirischen Disziplinen wie Soziologie und Psychologie an. Die Vorgehensweise, Dokumentation und das Ergebnis eines Experiments müssen so gestaltet sein, dass andere anhand der Dokumentation die Umgebung des Experiments, die Durchführung sowie das Ergebnis reproduzieren können müssen. Gleichzeitig muss das Experiment über eine möglichst hohe externe und interne Validität verfügen. Das bedeutet, dass die Ergebnisse möglichst generalisierbar sein sollen, dabei gleichzeitig aber auch die Kausalität der mit dem Experiment untersuchten Zusammenhänge klar bleiben muss.

Es existiert eine große Bandbreite an Literatur die beschreibt wie man als Wissenschaftler diesen Anforderungen gerecht werden kann. Einen guten Überblick für den Einstieg in die Praxis bieten Lazar, Feng und Hochheiser (2010). Deutlich umfangreicher sind die Werke von Dix u. a. (2003), Jacko (2012) und MacKenzie (2013).

Neben den dort beschriebenen Grundlagen gibt es beim Durchführen von Experimenten zur Messung von Texteingabeleistung noch zusätzliche Aspekte zu beachten.

Verschiedene Arten von Prototypen

Zum Definieren des Funktionsumfangs eines Produktes sowie zur Untersuchung, wie gut eine Funktion zum Lösen einer Aufgabe umgesetzt wurde, eignen sich Prototypen in verschiedenen Stufen der Umsetzungstreue. Diese ist ein Maß dafür, wie nah ein Prototyp an die verschiedenen Eigenschaften eines marktreifen Produktes herankommt. Dies kann zum Beispiel die Qualität der grafischen Umsetzung sein, die Vollständigkeit des Interaktionskonzepts oder auch die Verwendung eines anderen Mediums im Prototypen. Viele Erkenntnisse lassen sich bereits bei niedriger Umsetzungsgüte erzielen, zum Beispiel das Testen eines Arbeitsablaufs mit einem Papier-Prototypen oder einem Prototypen, der nur einen sehr eingeschränkten Funktionsumfang besitzt (Sauer, Seibel und Rüttinger 2010; Virzi, Sokolov und Karis 1996). Durch ein erkennbar unfertiges Aussehen des Prototypen kann auch dafür gesorgt werden, dass Testpersonen tatsächlich nur das zu untersuchende Konzept beurteilen und nicht etwa enttäuschte implizite Erwartungen an die Funktionen eines Produkts die Messergebnisse verfälschen. Dabei ist zu beachten, dass der Prototyp dem zu untersuchenden Konzept angemessen ist. Das bedeutet, dass die Testpersonen die impliziten und expliziten Eigenschaften eines Konzepts in ihrer Gesamtheit erfassen und beurteilen können.

Die Auswahl der Art des Prototypen bleibt letztlich dem Versuchsleiter überlassen, da es hier keine pauschal richtige Vorgehensweise gibt.

Aufgabenart

Texteingabe ist normalerweise kein Selbstzweck, sondern das Endergebnis eines mentalen Prozesses und dient der Fassung von Gedanken in Worten. Die Schwierigkeit bei der Messung von Texteingabeleistung besteht darin, wirklich die Texteingabe zu messen und nicht den vorhergehenden mentalen Prozess. Gibt man einer Versuchsperson die Aufgabe, einen Text von 1000 Zeichen zu verfassen, so wird die Person sich zunächst einen Text ausdenken. Dies kann bei unterschiedlichen Personen verschieden lange dauern; außerdem ist der entstehende Text mit hoher Wahrscheinlichkeit nicht vergleichbar: Wo eine Person durch das experimentelle Setting dazu angeregt wird, sich intellektuell zu verausgaben und komplizierte Fachbegriffe zu verwenden, mag eine andere Person einfach nur mit der Aufgabe fertig werden wollen und kurze, einfache Sätze erzeugen. Dies ergibt bei der Eingabe des Textes Unterschiede, die nicht auf die tatsächlichen Nutzungseigenschaften der untersuchten Eingabemethode zurückzuführen sind. Diese Einflüsse sind in der Messung nicht zu unterscheiden und können zu verfälschten Ergebnissen führen. Auch wenn diese Art der Aufgabenstellung dem realen Verwendungsszenario recht nahe kommt, ist sie für daher für quantitative Messungen nicht wirklich geeignet.

Oft wird in Experimenten auf eine andere, im alltäglichen Leben eher untypische Tätigkeit zurück gegriffen: Das Abschreiben eines vorgegebenen Textes. Diese Aufgabe bietet den Vorteil, dass der Versuchsleiter die Schwierigkeit und Länge des Textes genau kontrollieren kann. Damit die verwendeten Texte auch die spezifischen Eigenschaften bezüglich Wortlänge und Buchstabenverteilung der jeweiligen Sprache widerspiegeln, kann auf vorbereitete Textkataloge in Form eines Korpus oder von *Phrase Sets* zurückgegriffen werden (MacKenzie und Soukoreff 2003).

Gewichtung

Alle hier vorgestellten Aspekte bewegen sich in einem kontinuierlichen Spektrum. Die Entscheidung, welche Texteingabetechnik für ein gegebenes Szenario am besten geeignet ist, hängt von der Gewichtung dieser Faktoren ab. Dabei existiert keine „goldene Regel“, wie diese Gewichtung gefunden werden kann. Idealerweise werden Nutzertests verwendet, um eine gute Kombination an Kriterien zu finden. Aber auch die Erfahrung des Entwicklers oder UX-Designers kann sehr hilfreich sein, um eine gut funktionierende Lösung zu finden.

2.2.4 Messung von Texteingabeleistung

Um die „Leistung“ verschiedener Texteingabelösungen objektiv miteinander vergleichen zu können, sind messbare Kriterien notwendig. Allerdings ist Leistung im Kontext der Texteingabe nicht eindeutig definiert: Bezieht sie sich auf die Geschwindigkeit,

die Anzahl eingegebener Zeichen pro Zeiteinheit? Oder auf die Zahl der Fehler bei der Eingabe eines Textes? Oder auf die Effizienz der Eingabelösung, darauf wie viel Interaktion notwendig ist um einen bestimmten Text einzugeben – welche wieder von der Anzahl an Fehlern abhängt?

Dieser Abschnitt stellt übliche Parameter zur Beurteilung von Texteingabesystem vor und beschreibt, was bei der Messung wichtig ist. Das Ziel ist es, ein Grundverständnis für die folgenden Inhalte herzustellen. Für einen deutlich detaillierteren Überblick von Texteingabe in mobilen Kontexten sei auf eine Veröffentlichung von MacKenzie und Soukoreff (2002) verwiesen. Ebenfalls von MacKenzie und Tanaka-Ishii (2010) gibt es ein empfehlenswertes Buch, das sich komplett mit Texteingabe beschäftigt und relevante Aufsätze verschiedener Autoren enthält.

Metriken der Texteingabeleistung

Es gibt verschiedene Metriken für Texteingabe, die unterschiedliche Aspekte der Güte einer Texteingabelösung bewerten. Im folgenden werden Bedeutung und Berechnung der von MacKenzie und Soukoreff (2002) vorgeschlagenen Metriken Geschwindigkeit, Fehlerrate, Effizienz und Erlernbarkeit kurz anhand zweier Beispiele vorgestellt.

Beispiel 1: Einfache Software-Tastatur Ein Nutzer verwendet eine Bildschirmtastatur, um folgenden Satz einzugeben:

```
1 Texteingabe lässt sich anhand verschiedener Kriterien messen.
```

Dieser Satz besteht aus 61 Zeichen, dabei sind Leerzeichen und Satzzeichen inkludiert. In diesem Versuch betätigte der Nutzer insgesamt jedoch 78 mal eine Taste, da ihm bei der Eingabe Fehler unterliefen und er diese ausbesserte. Insgesamt benötigte er dafür 86,6 s.

Beispiel 2: Vorhersagende Software-Tastatur Der Nutzer schreibt bei Verwendung eines lernenden Texteingabesystems häufig die Grußformel

```
1 Mit freundlichen Grüßen
```

Sie ist 23 Zeichen lang, allerdings müssen diese Zeichen nicht einzeln eingeben werden: Der Nutzer bekommt nach der Eingabe des ersten Buchstabens `M` nacheinander die Worte `Mit`, `freundlichen` und `Grüßen` vorgeschlagen (vgl. Abb. 2.6). Mit je einer Berührung kann er diese Vorschläge akzeptieren, er berührt insgesamt nur vier mal den Bildschirm. Für diese Eingaben benötigt er nur 4,4 s.

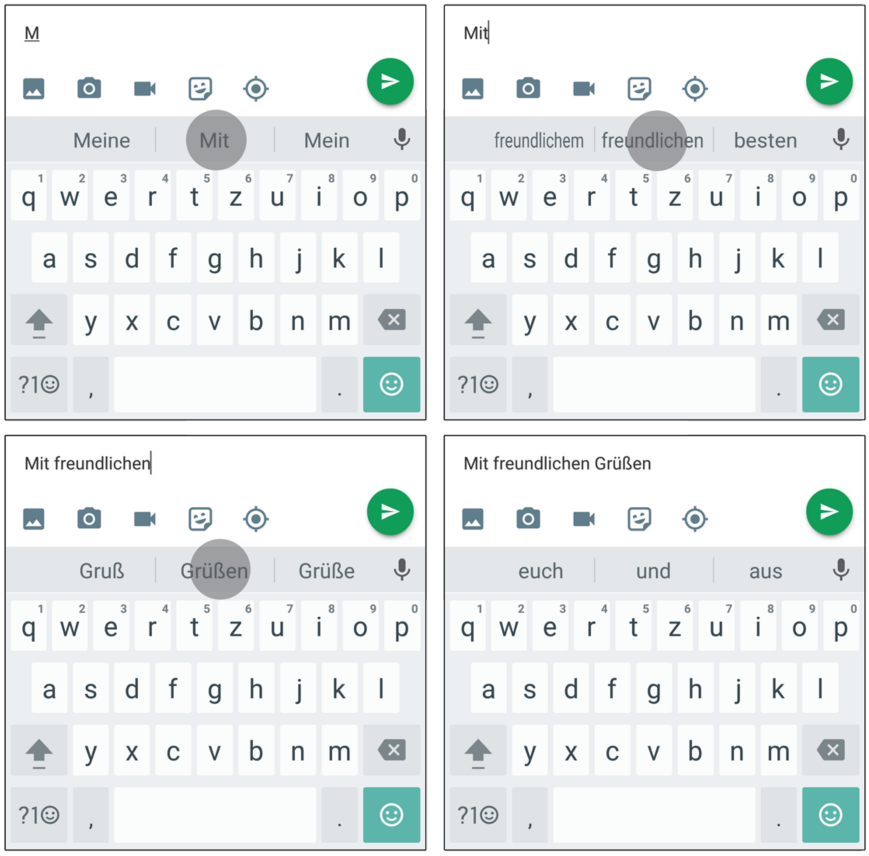


Abbildung 2.6: Bedienung einer vorhersagenden Tastatur nach initialer Eingabe des Buchstabens „M“. Die grauen Kreise zeigen die Berührung des Bildschirms durch den Nutzer.

Geschwindigkeit

Words per minute, wpm Die Geschwindigkeit einer Texteingabe wird üblicherweise in „Worten pro Minute“ gemessen (engl.: *words per minute*, abgekürzt: *wpm*). Im umgangssprachlichen Gebrauch würde man sagen, dass der Text aus Beispiel 1 aus sieben Worten besteht. Da sich die Durchschnittslänge eines Wortes sowie die Frequenz kurzer Füllwörter jedoch je nach Sprache unterscheiden kann und Ergebnisse aus verschiedenen Ländern damit nicht mehr vergleichbar wären, wurde die Wortlänge auf fünf Zeichen standardisiert. Um die *wpm* eines Textes zu berechnen wird einfach die Anzahl der Zeichen eines Textes erst durch fünf, anschließend durch die für die Eingabe notwendige Dauer in Minuten geteilt.

***wpm* in Beispiel 1 und 2** Für die Berechnung der *wpm* wird die Anzahl der Zeichen des Textes durch fünf geteilt. Er besteht aus $61/5 = 12,2$ Worten. Damit ergibt sich folgende Rechnung:

$$\frac{61\,\text{Zeichen}}{5} = 12{,}2\,\text{Worte}$$

$$\frac{86{,}6\,\text{s}}{60\,\text{s/min}} \approx 1{,}44\,\text{min}$$

$$\frac{12{,}2\,\text{Worte}}{1{,}44\,\text{min}} \approx 8{,}47\,\text{wpm}$$

Die Berechnung der *words per minute* für Beispiel 2 erfolgt analog und ergibt einen Wert von 76,7 wpm.

Keystrokes per second, ksps Eine weitere Geschwindigkeits-Metrik berechnet sich einfach aus der Anzahl der Tastendrücke pro Sekunde (engl.: *keystrokes per second*, abgekürzt *ksps*). Sie kann als Indikator verwendet werden, wie ergonomisch und erreichbar die Tasten einer Tastatur angeordnet sind.

***ksps* in Beispiel 1 und 2** Der Nutzer erzielte im ersten Beispiel 0,9 Tastendrücke pro Sekunde:

$$\frac{78\,\text{keystrokes}}{86,6\text{s}} = 0{,}90\,\text{ksps}$$

Diese Metrik ist allerdings nur dort sinnvoll einsetzbar, wo tatsächlich pro Zeichen mindestens ein Tastendruck notwendig ist. Bei vorhersagenden System verliert der Wert

an Aussagekraft. Im zweiten Beispiel wird mit nur vier Interaktionsvorgängen ein Text von 23 Zeichen eingegeben. Daraus ergibt sich eine Geschwindigkeit von 76,7 wpm bei einer Rate von 0,91 ksps. Eine Gegenüberstellung mit den Werten aus dem ersten Beispiel zeigt, dass beinahe identische ksps-Werte sehr unterschiedliche wpm-Werte erzeugen können: 8,47 wpm zu 76,7 wpm. Eine sinnvolle Vergleichbarkeit ist damit nicht mehr gegeben.

Fehlerrate

Die Fehlerrate gibt an, wie viele Fehler dem Nutzer bei der Eingabe von Text unterlaufen. Sie ist ein Maß dafür, wie gut sich eine Texteingabetechnik nutzen lässt. Dabei kann der Nutzer allerdings fast immer einen Kompromiss aus Geschwindigkeit und Fehlerrate wählen: Eine genauere Nutzung des Eingabesystems verringert in der Regel sowohl die Fehlerrate als auch die Geschwindigkeit.

Bei der Messung der Fehlerrate müssen zwei Klassen von Fehlern unterschieden werden: Solche, die der Nutzer bemerkt und korrigiert und solche, die der Nutzer nicht korrigiert und die sich daher im fertigen Text noch auffinden lassen. Korrigierte Fehler lassen sich nicht direkt messen, wenn nur der vollendete Text zur Verfügung steht. Indirekt kann man Fehler aus der Anzahl der Editiervorgänge ableiten, wenn die prinzipielle Funktionsweise des Eingabesystems bekannt ist. Dabei kann aber nicht immer unterschieden werden, ob es sich beispielsweise um einen einzelnen Fehler handelte, der erst spät bemerkt wurde und daher erst nach mehrfacher Verwendung der Pfeiltasten korrigiert werden konnte oder ob mehrere Fehler auftraten, die jeweils sofort bemerkt und korrigiert wurden. Es ist daher sinnvoll, jeden einzelnen Editiervorgang des Nutzers zu protokollieren. Dann lassen sich Fehlerraten ohne Probleme auch bei korrigierten Texten berechnen.

Als Maß für nicht korrigierte Fehler kann die sog. „Levenshtein-Distanz“ (Levenshtein 1966) verwendet werden. Sie beschreibt, wie viele Einfüge-, Ersetzungs- und Löschvorgänge notwendig sind, um eine Zeichenkette in eine andere Zeichenkette zu überführen. Mit ihr kann gut verglichen werden, wie sehr sich eine fehlerhafte von der gewünschten Eingabe unterscheidet.

Fehlerrate in Beispiel 1 und 2 Zu den beiden Beispielen lassen sich keine Fehlerraten berechnen, da der Text am Ende jeweils genau der Vorgabe entsprach. Da in Beispiel 1 allerdings deutlich mehr Tasten betätigt wurden (insgesamt 78) als für eine fehlerfreie Eingabe notwendig wären (63, davon 61 für die Zeichen und zwei Mal die Umschalttaste für die beiden Großbuchstaben), kann man davon ausgehen, dass Fehler auftraten und korrigiert wurden.

Effizienz

Die Effizienz einer Texteingabe lässt sich daran messen, wie viele Zeichen pro Interaktion ausgegeben werden. Sie wird in der Einheit *keystrokes per character* (abgekürzt *kspc*) angegeben. Bei einer klassischen Tastatur wird pro Tastendruck genau ein Zeichen ausgegeben, was einer Effizienz von 1 kspc entspricht.

Die Effizienz sinkt, wenn mehr Interaktionsvorgänge nötig sind als Zeichen ausgegeben werden sollen; dabei wird der kspc-Wert größer. Typischerweise ist das bei der Eingabe von Großbuchstaben der Fall, da ein zusätzlicher Tastendruck der Umschalt-Taste nötig ist.

Wenn der Nutzer bei der Eingabe Fehler macht und diese dann korrigieren muss, kommt es zu einer Verschlechterung der Effizienz. Selbst bei sofortigem Bemerken eines falsch eingegebenen Buchstabens muss einmal die Löschen-Taste betätigt werden, gefolgt vom korrekten Buchstaben. Statt eines Tastendrucks wurden für diesen Buchstaben nun drei benötigt. Falls der Vertipper schon weiter zurück liegt, muss zum Erreichen der Fehlerstelle die Lösch- oder eine Pfeiltaste mehrfach betätigt werden, sodass die Effizienz durch einen einzelnen Fehler deutlich sinken kann.

Eine Steigerung der Effizienz wird durch einen kleineren *kspc*-Wert angezeigt und kann z.B. durch eine Wortvervollständigung bzw. -vorhersage erreicht werden. Dabei schlägt das System bei Eingabe von Wortanfängen gleich ganze Wörter vor, mit denen das eingegebene Präfix komplettiert werden kann. Floskeln oder häufig verwendete Wort-Folgen können ebenso gelernt und vorgeschlagen werden.

Beispiel Für das Beispiel 1 berechnet sich die Effizienz in *keystrokes per character* wie folgt:

$$\frac{78\,\text{keystrokes}}{61\,\text{characters}} \approx 1{,}28\,\text{kspc}$$

Für ein vorhersagendes System sei noch einmal das Beispiel aus 2.2.4 aufgegriffen: Ein Nutzer gibt mit einem vorhersagenden System die Floskel `Mit freundlichen Grüßen` ein. Nach Eingabe des `M` wird ihm die Vervollständigung auf `Mit` angeboten, danach die kompletten Worte `freundlichen` und `Grüßen`. Mit insgesamt vier Interaktion kann er so 23 Zeichen eingeben, das ergibt in kspc:

$$\frac{4\,\text{keystrokes}}{23\,\text{characters}} \approx 0{,}17\,\text{kspc}$$

Erlernbarkeit

Unter der Erlernbarkeit eines Texteingabesystems werden verschiedene Aspekte zusammengefasst. Zum einen wird untersucht, wie lange ein neuer Nutzer dieses Systems braucht, um es zufriedenstellend nutzen zu können. Zum anderen geht es auch darum, wie schnell die Nutzung des Systems „in Fleisch und Blut“ übergeht, sodass der Nutzer auch nach einer Nutzungspause noch gut mit dem System zurecht kommt. Untersucht werden kann Erlernbarkeit, indem Testpersonen mehrere Male, jedoch mit größerem zeitlichen Abstand, gebeten werden, ein Experiment zur Leistungsmessung durchzuführen. Aus dem zeitlichen Verlauf der Eingabeleistung lässt sich dann ableiten, welches der verglichenen Eingabesysteme leichter erlernbar ist.

2.2.5 Optimierungen von Bildschirmtastaturen

In Folge des Smartphone-Booms der letzten Jahre wurde die Forschung an Texteingabe mit Bildschirmtastaturen ein wichtiges Thema.

Anpassung der Tastengröße

Eines der offensichtlichen Probleme bei der Nutzung einer Bildschirmtastatur auf einem mobilen Gerät ist die geringe Größe der dargestellten Tasten.

Fitts' Gesetz bei Tastaturen Die einzelnen Tasten einer üblichen PC-Tastatur haben eine Fläche von $2\,\mathrm{cm}^2$ bis $4\,\mathrm{cm}^2$, je nach Layout der Tastatur befinden sich über 100 individuell beschriftete Tasten eng nebeneinander. Um sie sicher bedienen zu können, ist ein hohes Maß an feinmotorischer Kontrolle notwendig. Noch deutlich kleiner sind die Tasten bei der Verwendung einer Bildschirmtastatur auf einem mobilen Gerät, die Tasten haben hier meist nur wenige Millimeter Kantenlänge. Das Treffen solch kleiner Ziele ist nach Fitts (vgl. Abschnitt 2.1.2) eine herausfordernde Aufgabe. Sein Gesetz trifft allerdings nur Aussagen über Zeigeaufgaben, die mit einem einzelnen Zeigeobjekt vorgenommen werden. Während geübte Nutzer klassische Tastaturen mit bis zu zehn Fingern gleichzeitig bedienen, sodass Fitts' Gesetz nur begrenzt gilt, werden Bildschirmtastaturen oft mit einzelnen Fingern oder einem Stylus bedient. In diesem Fall gehen sowohl die Größe W_s als auch die Entfernung A zwischen zwei aufeinander folgend betätigten Tasten in Fitts' Gesetz ein. Damit gilt für solche Tastaturen:

- Kleinere Tasten haben eine langsamere Eingabegeschwindigkeit zur Folge.
- Je weiter zwei Tasten voneinander entfernt sind, desto länger dauert ihre sequentielle Betätigung.

Es ergeben sich zwei einfache Gestaltungsregeln für Bildschirmtastaturen: Tasten sollten so groß wie möglich sein. Abstände zwischen Buchstaben sollten klein sein. Vor allem der zweite Punkte ist wichtig, da Buchstabenhäufigkeiten und -abfolgen in menschlicher Sprache nicht gleich verteilt sind. Bestimmte Kombinationen von Buchstaben (sogenannte Di-, Tri-, Tetra- oder Pentagramme, je nach Anzahl der Buchstaben (Mayzner und Tresselt 1965)) treten mit höherer Wahrscheinlichkeit auf als andere und werden daher auch häufiger nacheinander eingegeben. Idealerweise wird dies bei der Gestaltung der Tastatur berücksichtigt, sodass sich Buchstaben solcher Folgen nahe beieinander befinden, um die Eingabezeit zu verringern.

Nutzung im Querformat Gerade bei Mobilgeräten ist es oft möglich, das Gerät aus dem Hoch- in das Querformat zu drehen, um eine deutlich breitere und besser bedienbare virtuelle Tastatur angezeigt zu bekommen. Nachteilig wirkt sich dabei aus, dass die für die Darstellung des eigentlichen Programms verbleibende Fläche stark reduziert wird und ein sehr breites Seitenverhältnis hat. Dies kann sogar dazu führen, dass das mit der Eingabe zu füllende Textfeld im Querformat nicht mehr sichtbar ist und dem Nutzer so der mentale Kontext verloren geht.

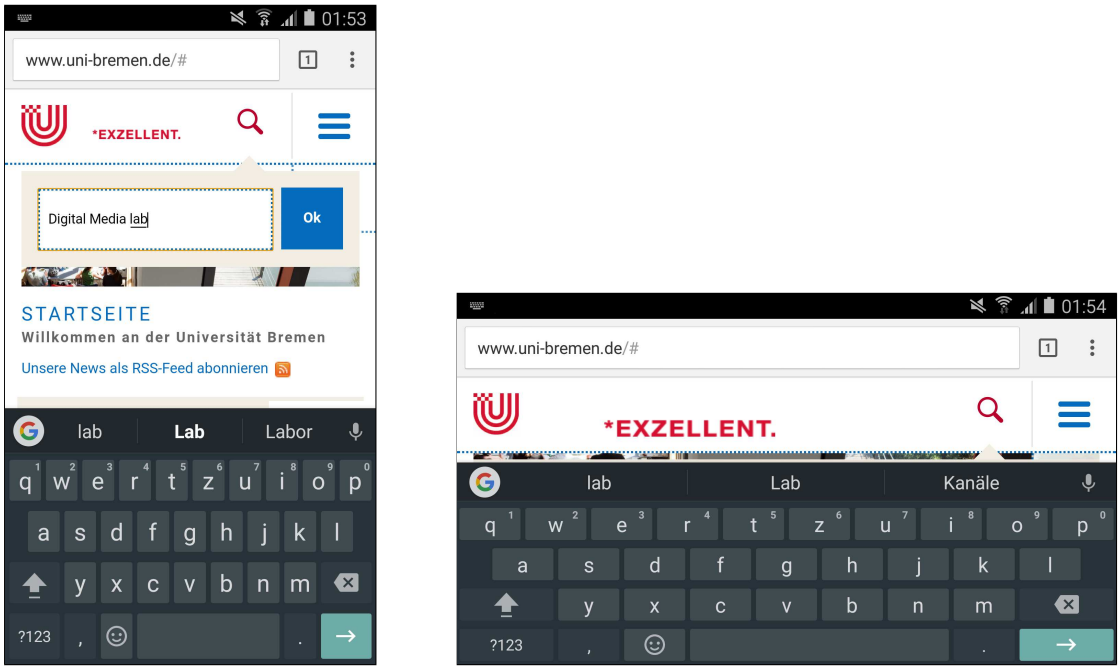
Anpassung der Trefferzonen Um Fehlbedienungen aufgrund zu kleiner Tasten zu verringern, ist eine Vergrößerung der Tasten die offensichtliche Wahl. Wenn der vorhandene Raum bereits gleichmäßig ausgefüllt wird, können offensichtlich nicht alle Tasten vergrößert werden.

Eine Vergrößerung nur einzelner Tasten auf Basis statistischer Auswertungen von Digrammen untersuchten Al Faraj, Mojahid und Vigouroux (2009). Dabei wurden nach Eingabe des ersten Buchstaben eines Digramms die vier Buchstabentasten mit der höchsten Folgewahrscheinlichkeit vergrößert dargestellt, was nach Fitts' Gesetz eine höhere Eingabegeschwindigkeit und Genauigkeit zur Folge haben sollte. In der Tat wurde experimentell eine Steigerung der Eingabeleistung um 25% gegenüber einer normalen, nicht vergrößernden Tastatur festgestellt.

Gunawardana et al. untersuchten eine Tastatur, die Ein- und Ausgabe-Funktion des berührungsempfindlichen Bildschirms getrennt nutzte. Bei unveränderter grafischer Darstellung wurden anhand statistischer Modelle die Trefferzonen von Tasten mit hoher Folge-Wahrscheinlichkeit vergrößert. Mit diesem Ansatz gelang es, die Anzahl falsch eingegebener Tasten von ca. 9,5 % auf ca. 7,5 % zu senken (Gunawardana, Paek und Meek 2010).

Vorhersage und automatische Korrektur

Ebenfalls auf statistischen Verfahren aufbauend ist der Ansatz, selbst bei fehlerhafter Eingabe die Worte zu erkennen, die der Nutzer tatsächlich eingeben wollte. So ist eine



(a) Ein Eingabefeld auf einer Webseite

(b) Im Querformat ist das Eingabefeld nicht mehr sichtbar.

Abbildung 2.7: Fehlender Kontext bei der Verwendung eines Suchfelds auf einer Webseite. Im Hochformat sind Suchfeld und Tastatur gleichzeitig sichtbar. Beim Rotieren des Geräts schalten Internet-Browser und Bildschirmtastatur ins Querformat um, das Eingabefeld ist jedoch ohne Interaktion nicht mehr sichtbar. (Abbildung sind im gleichen Maßstab)

Autokorrektur möglich, die im Idealfall den Nutzer davon befreit, falsch eingegebene Buchstaben selber korrigieren zu müssen.

Beispiel für automatische Korrektur Beim Eingeben eines Wortes unterläuft dem Nutzer ein Fehler, so dass die Eingabe `Tasratur` lautet. Aufgrund des geringen Editierabstands zwischen den beiden Worten und der räumlichen Nähe der Buchstaben `r` und `t` kann nun der Schluss gezogen werden, dass vermutlich `Tastatur` gemeint war und der Fehler kann automatisch korrigiert werden.

Wenn die Vorhersage nicht eindeutig ist, werden dem Nutzer die verschiedenen Möglichkeiten zur Auswahl angeboten. Da es immer sein kann, dass das eingegebene Wort im statistischen Modell nicht vorkommt und nur daher nicht als korrekt erkannt wird, muss der Nutzer auch die ursprüngliche Eingabe auswählen können.

Das Verfahren kann ausgeweitet werden, sodass nicht nur das aktuell einzugebende Wort, sondern auch das im Text folgende Wort vorhergesagt und vorgeschlagen wird. Sollte diese Vorhersage korrekt sein, kann der Nutzer wie bereits beschrieben das vorgeschlagene Wort einfach antippen und so in seine Eingabe übernehmen (vgl. Abschnitt 2.2.4 und Abb. 2.6).

Gesten-Interfaces: Swiping statt Tapping

Auf Basis statistischer Modellierung der Eingabe ist es auch möglich, statt einzeln anzutippender Buchstaben Gesten zu verwenden. Bei dieser Eingabetechnik werden mittels einer Wischbewegung nacheinander die gewünschten Buchstaben auf der Tastatur überstrichen, so dass sich im Endeffekt eine auf dem Bildschirm gezeichnete Geste ergibt (vgl. Abb. 2.8). Diese Geste kann nun mit statistischen Verfahren mit gespeicherten Gesten abgeglichen werden und so eine Vorhersage für das gewünschte Wort getroffen werden (Kristensson und Zhai 2004; Zhai und Kristensson 2012). Bildschirmtastaturen dieser Art sind inzwischen z.B. unter den Namen *Swype* und *SwiftKey* auch für den Endnutzer verfügbar.

Abkehr von klassischen Tastatur-Layouts

Wie auch bei der Forschung zur Texteingabe an Desktop-PCs kann bei Bildschirmtastaturen untersucht werden, ob und welche Vorteile sich durch eine Änderung des Tastaturlayouts erzielen lassen. Dank der rein softwarebasierten Darstellung ist es sogar noch besser möglich, radikal andere Konzepte zu verfolgen.

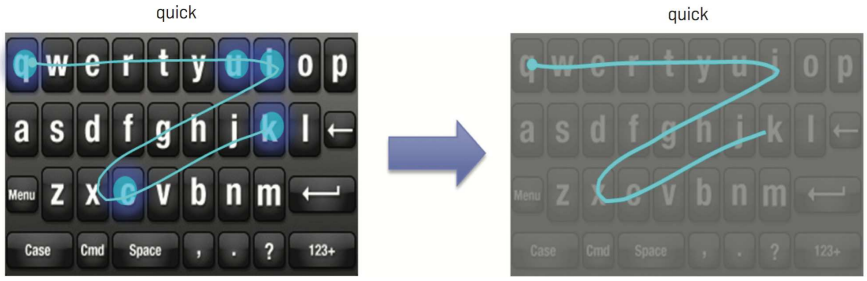


Abbildung 2.8: Eingabe des Wortes „quick“ mittels einer Geste durch Überstreichen der einzelnen Buchstaben des Wortes. Abbildung aus Zhai und Kristensson 2012

2.3 Hover Detection

Hover Detection (engl.) bedeutet wortwörtlich, in der Luft schwebende Gegenstände zu erkennen. Während der Beriff für die gegenwärtige und in dieser Arbeit gemeinte Interaktionsform im Zusammenhang mit Mensch-Maschine-Interaktion korrekt ist, durchlief das Verb „to hover“ in der Geschichte des Personal Computing eine Bedeutungsänderung.

2.3.1 Abgrenzung zum mausbasierten „Hovering“

Ursprünglich wurde der Begriff *hover* im Zusammenhang mit dem Mauszeiger verwendet: Man spricht von „hovering“, wenn sich der Mauszeiger zwar über einem Steuerelement befindet, dieses jedoch (noch) nicht durch einen Klick aktiviert wurde. Interaktive Steuerelemente können diese Information beispielsweise nutzen, um ihre grafische Darstellung zu ändern und so eine Aufforderung zur Interaktion darzustellen. Neben der Nutzung in Bibliotheken zur Darstellung grafischer Oberflächen fand der Begriff mit dem Aufstieg des World Wide Web weite Verbreitung im Zusammenhang mit der Sprache *Cascading Style Sheets* (CSS), die zur Beschreibung des Aussehens von Webseiten verwendet wird. Mittels sogenannter *Selektoren* kann hier definiert werden, welche Elemente der Webseite welches Aussehen haben sollen. Einer dieser Selektoren erlaubt unter dem Namen `:hover` die Definition der Darstellung von Webseiten-Elementen, die sich zwar unter dem Mauszeiger befinden, jedoch nicht angeklickt wurden.

Da ein Mauszeiger nur die abstrakte Darstellung einer Interaktionsform ist, handelt es sich dabei um ein reines Software-Phänomen. Es existiert kein tatsächlicher oder virtueller räumlicher Abstand zwischen den dem Steuerelement und dem Mauszeiger; die Tiefenstaffelung ist rein logisch.

Ohne Maus bzw. Mauszeiger steht jedoch keine kontinuierliche virtuelle Repräsentation des Zeigers mehr zur Verfügung, die für diese Art des „Hovering“ verwendet wer-

den könnte. Bei einem berührungsempfindlichen Bildschirm löst eine Berührung sofort eine Aktion aus, die Position der Interaktion steht ebenso erst in genau diesem Moment bereit.

2.3.2 Echte Hover Detection

Tatsächliche Hover Detection basiert auf kontaktloser, räumlicher Interaktion. Dafür ist es notwendig, die räumliche Position sowie oft auch die Ausrichtung eines zur Interaktion bestimmten Gegenstands in Relation zum Interaktionsziel erfassen zu können. Die im folgenden vorgestellten Techniken eignen sich primär für den Einsatz im Nahbereich. Die erzielten Reichweiten und Genauigkeiten variieren jedoch stark. Die Verfahren sind entweder bereits verfügbar oder werden in wissenschaftlichen und kommerziellen Veröffentlichungen beschrieben.

Elektromagnetische Verfahren

Wenn elektromagnetische Felder oder Wellen für die Abstands- und Positionsmessung verwendet werden, so spricht man von elektromagnetischen Verfahren.

Weit verbreitet ist die Verwendung eines E-Feldes, das durch einen leitenden Gegenstand gestört wird. Dies kann ein Finger sein, der aufgrund seines Wassergehaltes als leitend betrachtet werden kann. Diese Störung lässt sich messen und daraus die Position des Gegenstands berechnen. Angewandt wird diese Technik zur Berechnung des Berührungspunktes vor allem in kapazitativen Touchscreens.

Objekt- und Entfernungserkennung kann auch mit Wellen im Radio-Spektrum umgesetzt werden. Dieses Verfahren ist unter dem Namen *Radar* (*Radio Detection and Ranging*) bekannt. Während klassischerweise große Reichweiten zum Beispiel bei der Erkennung und Verfolgung von Flugzeugen gewünscht werden, eignen sich vor allem die hohen Frequenzen des verwendeten Spektrums (30MHz bis 130GHz) für die Anwendung im Nahbereich weniger Zentimeter bis Meter.

Kapazitative Sensoren in Touchpads, Touchscreens und Grafiktabletts Kapazitative Sensoren basieren auf der Messung der Ändung der maximalen Ladung eines Kondensators. Ein geladener Kondensator erzeugt ein elektrisches Feld. Wird in dieses Feld ein elektrischer Leiter eingebracht, so verändert sich das Feld und damit die Ladung des Kondensators. Diese Änderung ist in der Regel zwar sehr klein, kann aber dennoch gemessen werden. Wenn es sich nun um ein regelmäßiges Gitter von Kondensatoren handelt, so kann durch die Bestimmung der Ladungsänderung mehrerer benachbarter Kondensatoren die Lage des störenden Leiters bestimmt werden.

Die hier beschriebene kapazitative Methode wird bei modernen Touchpads und Touchscreens eingesetzt. Bei Touchpads handelt es sich um rechteckige Flächen mit Kantenlängen im Bereich mehrerer Zentimeter, die Berührung mit einem oder mehreren Fingern erkennen und lokalisieren können. Sie werden häufig in Notebooks als Mausersatz eingebaut. Touchscreens sind Bildschirme, die ebenfalls Berührungen durch Finger oder andere Gegenstände erkennen können. Sie werden in Smartphones verbaut.

Grafiktabletts funktionieren ebenfalls mit kapazitativer Sensorik. Sie sind Eingabegeräte, die eine große Zeichenfläche (bis DIN A3) bieten, auf der mit einem speziellen Stift sehr präzise gezeichnet werden kann. In der Spitze des Stifts ist vorne eine Spule eingebaut, über die der Stift induktiv mit Strom versorgt wird. Mit Hilfe der bekannten elektrischen Eigenschaften der Spule lässt sich die Position des Stiftes auf und über der Zeichenfläche sehr genau feststellen. Über einen Drucksensor hinter der Spitze des Stiftes lässt sich zudem der Anpressdruck bestimmen. Die mit dem Drucksensor und den Schaltern erfassten Daten werden kabellos an das Tablett übertragen.

Es gibt zwei verschiedene sensorische Verfahren, die auf Messungen von Kapazitäten basieren und beide in berührungsempfindlichen Bildschirmen zum Einsatz kommen: *Gegenkapazität* und *Eigenkapazität*. Um zu verstehen, welche Eigenschaften sie haben und wo sie Beschränkungen unterworfen sind, ist etwas technisches Verständnis notwendig.

Gegenkapazität (engl. *mutual capacitance*) In einer zwischen Deckglas und eigentlichem Bildschirm befindlichen Schicht wird ein Gitter dünner, voneinander isolierter horizontaler und vertikaler Leitungen erzeugt. Alle Elektroden sind einzeln an einen entsprechenden Mikrocontroller angeschlossen. Nun wird an die horizontalen Elektroden nacheinander eine Spannung angelegt. Durch den Spannungsunterschied entsteht zwischen der horizontalen und den sie kreuzenden vertikalen Elektroden jeweils eine Gegenkapazität, die sich wie ein Plattenkondensator verhält. Beim Laden dieser Kondensatoren fließt ein sehr geringer Strom, der gemessen werden kann und für alle vertikalen Elektroden gleich groß ist. Wird nun ein leitender, geerdeter Gegenstand in das E-Feld eines der aktiven Kondensatoren gebracht (zum Beispiel in dem mit dem Finger das Glas darüber berührt wird), so verändert sich das elektrische Feld durch den zusätzlichen Erdungspfad. Auf den vertikalen Elektroden verändert sich dadurch der Stromfluss. Durch Korrelation mit der gerade aktiven horizontalen Elektrode kann so die Position der Berührung bestimmt werden. Durch eine hohe Abtastfrequenz der horizontalen Elektroden lassen sich auch mehrere Berührungen gleichzeitig messen. Allerdings ist die Empfindlichkeit des Sensors aufgrund der geringen Kapazitäten der Kondensatoren im Gitter so gering, dass nur eine sehr geringe räumliche Reichweite besteht (Carey 2009). In der Praxis muss der Bildschirm berührt werden, damit ein Finger erkannt wird.

Eigenkapazität (engl. *self capacitance*) Alternativ kann an alle, sowohl die horizontalen als auch die vertikalen Elektroden des Gitters die gleiche Spannung angelegt werden. Die Elektroden bilden nun einen Plattenkondensator „mit der Erde“, man spricht von ihrer Eigenkapazität. Beim Abschalten der Spannung fließt die Ladung aus den Elektroden zurück und kann dabei gemessen werden. Auch hier ist sie für alle Leiter gleich groß. Wird nun ein Finger in die Nähe des Gitters gebracht, so bietet er einen zusätzlichen Erdungspfad und beim Abschalten der Spannung verändert sich die zurückfließende Ladung. Diese Änderung kann gemessen werden und erlaubt festzustellen, in der Nähe welcher horizontalen und vertikalen Elektrode sich der Finger befand. Ebenfalls möglich ist ein Rückschluss auf die Entfernung des Fingers zu den Elektroden. Mit dieser Technik ist allerdings nicht möglich, mehrere Berührungspunkte gleichzeitig zu erkennen, da die Korrelationen betroffener Leiter nicht eindeutig bestimmbar sind. Berührt der Nutzer beispielsweise die Oberfläche an den Punkten T_1 und T_2, so werden die darunter liegenden vertikalen Leitungen X_1 und X_2 sowie die horizontalen Leitungen Y_1 und Y_2 aktiviert. Welche Kombination allerdings die richtige ist, kann nicht festgestellt werden – anhand der festgestellten Leitungen wären wie in Abb. 2.9 dargestellt sowohl die Koordinatenpaare $T_1 = (X_1, Y_1)$ und $T_2 = (X_2, Y_2)$ als auch $T_1 = (X_1, Y_2)$ und $T_2 = (X_2, Y_1)$ möglich (Sony Mobile 2012).

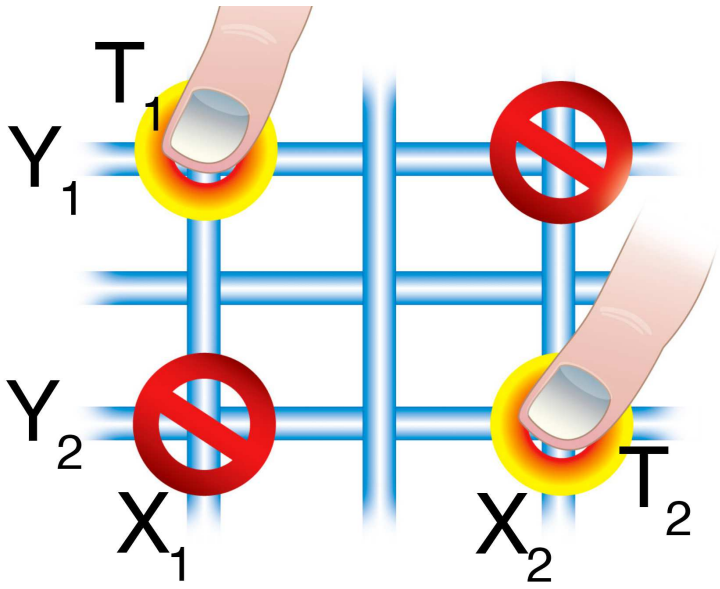


Abbildung 2.9: Auftreten von „Ghosting“ bei der mehreren Berührungen eines Touchscreens, der Eigenkapazität verwendet. (Abb. basierend auf 3M Touch Systems 2013)

Praktische Verwendung In so gut wie allen heute in Smartphones zum Einsatz kommenden Touchscreens wird Gegenkapazität eingesetzt, um mehrere gleichzeitige Berührungen des Bildschirms erkennen zu können. Da die Verdrahtung der Elektroden mit dem Mikrocontroller in beiden Fällen gleich ist, kann zusätzlich Eigenkapazität ein-

gesetzt werden, um Hover Detection zu ermöglichen. Dies wird allerdings nicht weitläufig angewendet und es wurden bisher nur wenige Geräte auf den Markt gebracht, die ein solches Feature implementierten.

Eine der ersten Erwähnungen von Hover Detection in Produkten für Endverbraucher findet sich in der technischen Dokumentation (Synaptics Inc. 2001) zu den von Synaptics hergestellten kapazitativen Touchpads. Dort ist die Rede von einem Z-Wert, der zusätzlich zu den normalen zweidimensionalen Koordinaten erfasst wird und die Stärke der gemessenen Kapazität des Fingers angibt. Bei einem Wertebereich von 0–255 gibt die Dokumentation an, dass ein Wert von $Z = 10$ für einen „Finger hovering near the pad surface“ steht (ebd., S. 8). Prinzipiell war mit einem solchen Touchpad schon vor Jahren Hover Detection möglich, auch wenn dem Autor in der Praxis keine Anwendungsfälle bekannt sind.

Sony *Floating Touch* Das erste Smartphone, das einen über dem Bildschirm schwebenden Finger erkennen konnte, wurde 2012 von Sony unter der Bezeichnung „Xperia Sola“ auf den Markt gebracht. Die Hover Detection wurde als „Floating Touch“ vermarktet.

In der Software des Geräts wurde die Hover Detection im Browser verwendet, um Maus-Hover-Effekte (vgl. 2.3.1) mit echter Hover Detection zu realisieren. Eine Schnittstelle für Entwickler existierte zum Zeitpunkt der Auslieferung nicht. Sie wurde zwar für eine spätere Betriebssystemversion versprochen (Sony Mobile 2012), jedoch nie veröffentlicht.

Samsung *AirView* Mit dem Samsung *Galaxy S4* kam im März 2013 das erste Gerät mit Hover Detection auf den Markt, das auch eine Schnittstelle für Programmierer für dieses Feature bot. Die Technik wurde unter der Bezeichnung *AirView* beworben und wie bei Sony in verschiedenen mitgelieferten Anwendungen zum Einsatz gebracht. Eine Unterstützung der klassischen Hover-Effekte (vgl. 2.3.1) existierte im Browser jedoch nicht, stattdessen wurde die Funktion zum Einblenden einer Lupe verwendet. Auch in den anderen Anwendungen wurde die Hover Detection nur genutzt, um vergrößerte oder Detail-Ansichten einzublenden.

Sowohl die Bezeichnung AirView als auch die Programmierschnittstelle werden von Samsung auch in den Geräten der *Note*-Reihe eingesetzt, die außer mit dem Finger mit einem speziellen Stift bedient werden können. Dieser *S-Pen* sollte nicht mit den häufig zu kaufenden Stylus verwechselt werden, welche die elektrischen Eigenschaften eines menschlichen Fingers simulieren, indem ein leitfähiges Pulver (z.B. Graphit) mit Gummi oder Silikon vermengt wird. Stattdessen ist wie bei einem Grafiktablett eine kleine, induktive Spule im Stift verbaut, mit der sich hohe Genauigkeiten erzielen lassen.

Die in dieser Dissertation verwendeten Prototypen wurden mit AirView umgesetzt. Die spezifischen Eigenschaften von AirView werden in Abschnitt 3.3 anhand geeigneter Experimente untersucht, die Grundlagen der Programmierung von AirView werden im Anhang A.1 vorgestellt.

Abbildung 2.10: Verwendung von *AirView* in der Fotogalerie auf einem Samsung Galaxy S4: Wenn ein Finger über einem Vorschaubild schwebt, so wird ein größeres Vorschaubild eingeblendet.

Nahbereichsradar: Google *Project Soli* Ein sehr vielversprechender Ansatz zur Erkennung räumlicher Gesten wird im Moment bei Google entwickelt. *Project Soli* basiert auf der Erkennung von Handgesten mit Radar im ISM-Band für Anwendungen der Industrie, Wissenschaft und Medizin bei einer Frequenz von 60 GHz (Google Inc. 2015). Die räumliche Auflösung ist nicht hoch genug, um feine Finger-Gesten zu erkennen. Stattdessen werden Machine-Learning-Verfahren eingesetzt, um in den empfangenen Daten die ausgeführten Gesten erkennen zu können. Google verspricht eine sehr hohe Aktualisierungsrate von 100 Hz bis hin zu 10 KHz.

Nach seiner Vorstellung 2015 sind zwar nicht viele weitere Informationen über das Project Soli bekannt geworden, auf der Konferenz *Google I/O '16* stellten einige Hersteller jedoch schon prototypische Geräte vor. Darunter waren eine Smartwatch und ein Lautsprecher, die jeweils mit Gesten gesteuert werden konnten. Generell soll Projekt Soli in tragbaren Systemen, Telefonen, Computern, Autos und im Internet der Dinge zum Einsatz kommen. Ein offizielles Erscheinungsdatum steht noch nicht fest.

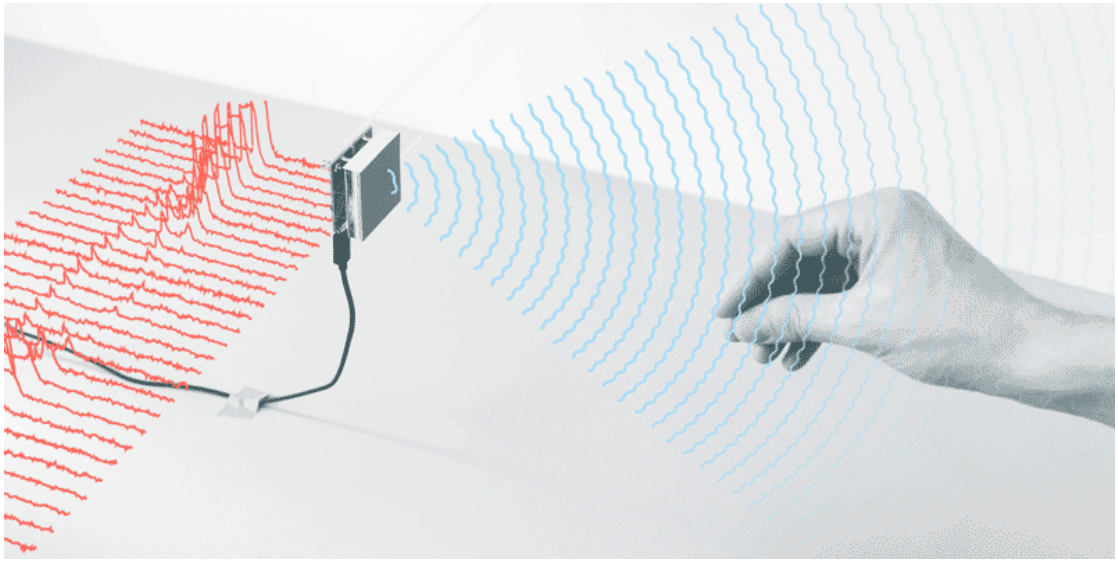
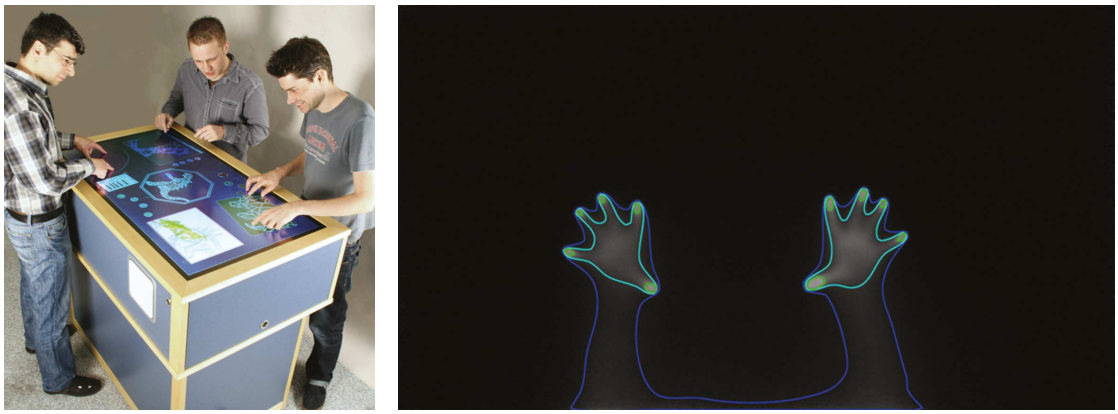


Abbildung 2.11: Konzept-Grafik für *Project Soli* (Abb. aus Google Inc. 2015)

Optische Verfahren

Eine Alternative zu den elektromagnetischen Verfahren stellen optische Verfahren dar. Sie fangen das von einem Objekt ausgestrahlte oder reflektierte Licht mit einem Sensor auf. Aus den Sensordaten werden die gewünschten Daten mit Computer-Vision-Verfahren extrahiert. Oft wird mit Licht im Infrarot-Spektrum gearbeitet. In der Praxis lassen sich Genauigkeiten im Zentimeterbereich bei einer Reichweite von mehreren Metern realisieren.

Multitouch-Tische Ein Beispiel für den Einsatz optischer Verfahren sind Multitouch-Tische. Diese bestehen im Prinzip aus einer vertikal montierten, halbdurchsichtigen Projektionsoberfläche, die einen doppelten Nutzen hat: Zum einen wird mit einem normalen Projektor ein Bild auf die Rückseite projiziert, das von oben sichtbar ist. Zum anderen wird die Rückseite mit Infrarot-Licht beleuchtet, das für einen Menschen nicht sichtbar ist. Werden nun Finger oder andere Gegenstände in die Nähe der Oberfläche gebracht, so kann das Infrarotlicht mittels *frustrated total internal reflection* (FTIR) (Han 2005) aus der Projektionsfläche austreten, an den Gegenstand abprallen und von einer ebenfalls unter der Projektionsfläche angebrachten Videokamera erkannt werden (vgl. Abb. 2.12b). Anhand des Videobildes können die Position und die Art der Reflektion, etwa Finger oder spezielle Marker (Kaltenbrunner und Bencina 2007), erkannt werden. Theoretisch sind räumliche Auflösung und die Anzahl der gleichzeitig erkennbaren Berührungen nur von der Auflösung der Infrarot-Kamera begrenzt. Praktisch werden Berührungen auf den Zentimeter genau erkannt und je nach System eine zweistellige Anzahl gleichzeitiger Berührungen unterstützt.



(a) Ein Multitouch-Tisch (b) Erkennung der Hände und Arme durch FTIR

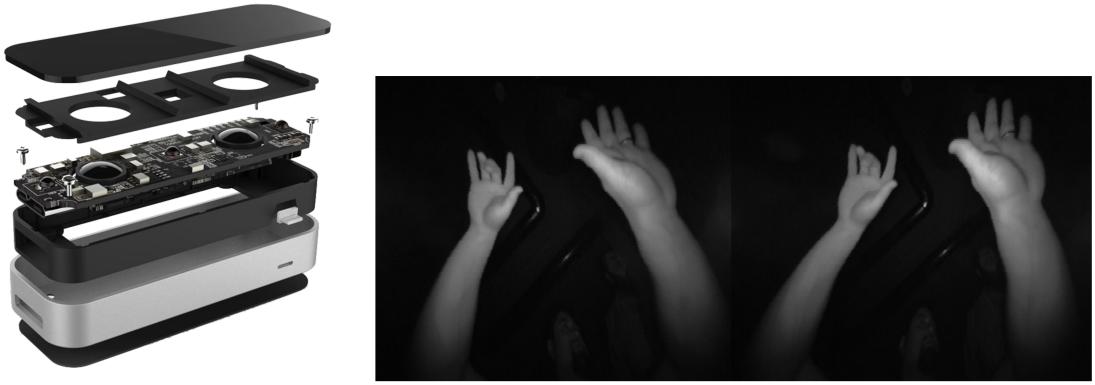
Abbildung 2.12: Nutzung von *frustrated total internal reflection* (FTIR) bei einem Multitouch-Tisch (Abb. aus Teichert u. a. 2010)

Leap Motion Der *Leap-Motion*-Controller der gleichnamigen Firma arbeitet ebenfalls mit reflektiertem Infrarotlicht. Allerdings ist dieses Gerät darauf ausgelegt, Objekte im dreidimensionalen Raum zu erkennen. In einem über USB an einen PC angeschlossenen Gehäuse (ca. 1 cm × 2 cm × 7 cm groß) sind mehrere Infrarot-LEDs enthalten sowie eine Stereo-Kamera mit einem Öffnungswinkel von ca. 120° × 150°. Der davon erfasste Bereich ist geformt wie eine umgedrehte Pyramide und reicht bis zu einer Höhe von 80 cm. In dem aufgenommenen Stereobild können mit Hilfe von Verfahren zum maschinellen Sehen die Positionen der Arme, Hände und sogar Finger erkannt werden. Über eine für mehrere Betriebssysteme und Programmiersprachen verfügbare Bibliothek können daraus Skelett-Daten errechnet werden. Die erzielten Genauigkeiten bewegen sich im Millimeterbereich. Da es sich um ein rein optisches Verfahren handelt, können Überdeckungen im praktischen Betrieb ein größeres Problem darstellen.

Microsoft *Kinect* Weit verbreitet ist der *Kinect*-Controller der Firma Microsoft, der als Zubehör für die Spiele-Konsolen *X-Box 360* und ihren Nachfolger *X-Box One* verfügbar ist.

Die erste Version von 2010 arbeitete mit einem Infrarot-Laser, der ein pseudozufälliges Punktmuster in den Raum projizierte. Die Reflektion des Musters wurde mit einer Kamera erfasst. Anhand der Verzerrung des Musters konnte der Controller berechnen, in welcher Distanz und von welcher Geometrie es reflektiert wurde (Lau 2013).

Die zusammen mit der *Xbox One* erschienene zweite Version des Controllers arbeitet mit Hilfe einer *time-of-flight*-Kamera. Bei dieser Technik wird ein Punkt im Raum mit einem gepulsten Laser beleuchtet, danach wird gemessen wie lange es dauert bis das



(a) Bestandteile des Controllers (b) Beispiel einer Stereoaufnahme

Abbildung 2.13: Der *Leap-Motion*-Controller (Abb. aus Colgan 2014)

(a) Kinect von 2010 (Abb. von Amos 2011) (b) Kinect von 2014 (Abb. von Amos 2014)

Abbildung 2.14: Die beiden Versionen der Microsoft *Kinect*

reflektierte Licht beim Sensor eintrifft. Aus der zeitlichen Differenz lässt sich zusammen mit dem Abstrahlwinkel des Lasers berechnen, wo die Reflektion stattfand (Lau 2013).

Für die Kinect-Controller steht ebenfalls ein SDK zur Verfügung, mit dessen Hilfe sie auch an PCs verwendet werden können.

Akustische Verfahren

Mit Ultraschall lässt sich eine im Vergleich einfache Entfernungsmessung realisieren, die bei einer Reichweite von bis zu 40 cm eine Genauigkeit im Millimeterbereich erzielen kann (Migatron Corp. 2016). Allerdings beeinflussen sich mehrere Sensoren gleichzeitig, so dass eine räumliche Hover Detection nicht umsetzbar ist.

Einordnung der Techniken

Für die Erkennung der räumlichen Position einer Person existieren mehrere Techniken, die verschiedene physikalische Phänomene ausnutzen. Allerdings variieren wie in Tabelle 2.1 dargestellt Genauigkeit und Randbedingungen teilweise erheblich. Für die in dieser Arbeit untersuchte Texteingabe auf einem mobilen Gerät eigenen sich akustische Verfahren nicht, da sie zu grobe Daten liefern. Ein optisches Erkennen könnte bei entsprechender Genauigkeit funktionieren, allerdings darf aufgrund des mobilen Einsatzzwecks ein sich stetig ändernder Hintergrund kein Problem darstellen. Ein Liniengitter wie bei der ursprünglichen Kinect ist damit aus dem Rennen, eine Erkennung auf Basis von Stereobildern wie bei der Leap Motion oder *Time-of-flight*-Daten wie bei der neuen Kinect hingegen wäre geeignet. Gegen beide Verfahren spricht allerdings, dass sie auf Verwendung einer Kamera basieren, die in mobilen Kontext nur im zu bedienenden Gerät selber montiert werden kann. Da der Sichtkegel einer Kamera über der Linse und damit direkt über dem Gerät die geringste Ausdehnung hat, ist eine Erkennung im Nahbereich über dem Bildschirm nur sehr eingeschränkt möglich.

Es bleiben die elektromagnetischen Verfahren: Nahbereichsradar erscheint vielversprechend, dürfte jedoch ähnliche Probleme mit dem Sichtwinkel haben wie optische Verfahren. Zudem existiert hier noch keine marktreife Lösung. Direkt verfügbar ist nur die Erkennung über kapazitative Messungen, wie sie bereits in einigen Smartphones verbaut und verkauft wird. Sie eignet sich daher als einzige Technik für die weitere Untersuchung in dieser Arbeit.

2.3.3 Forschung zu berührungsloser Interaktion

Hover Detection wird verwendet, um Hände oder Gegenstände *über* dem Eingabegerät zu erkennen. Prinzipiell ist die Anwendung natürlich jedoch nicht auf diese eine Richtung beschränkt. Je nach Anwendungsfall kann die Erkennung in beliebigen Richtungen um das Gerät herum stattfinden (engl.: *around-the-device interaction*). Im Folgenden sind einige Beispiele hierfür beschrieben:

Von Grossman u. a. (2006) wurden mit einem Stylus über einem Bildschirm durchgeführte Gesten als zusätzliche Eingabemodalität untersucht. Die Gesten wurden mit dem über dem Bildschirm schwebenden Stift ausgeführt und mit einer Berührung des Bildschirms abgeschlossen. Nutzer konnten die vorgestellten, einfachen Gesten schnell erlernen und zuverlässig ausführen. Diese Art der Eingabe erwies sich als besonders geeignet, wenn damit Aktionen ausgelöst werden konnten, für die ansonsten das Betätigen eines Knopfes oder einer räumlich entfernten Schaltfläche nötig gewesen wäre.

Eine Nutzung des physikalischen Raums neben dem Eingabegerät wurde von Butler, Izadi und Hodges (2008) untersucht. In ihrem als *Side-Sight* bezeichneten Konzept wird

Verfahren	Art der Erkennung	Reichweite	Genauigkeit
Kapazitative Sensoren	elektromagnetisch	wenige Zentimeter	hoch, Millimeter bis Zentimeter
Nahbereichsradar	elektromagnetisch	weniger als ein Meter	sehr hoch, Millimeter
frustrated total internal reflection	optisch	wenige Zentimeter	hoch, Millimeter bis Zentimeter
Stereokamera	optisch	weniger als ein Meter	hoch, Millimeter bis Zentimeter
Muster-basiert	optisch	mehrere Meter	mittel, mehrere Zentimeter
time of flight	optisch	mehrere Meter	mittel, mehrere Zentimeter
Ultraschall	akustisch	weniger als ein Meter	mittel, mehrere Zentimeter

Tabelle 2.1: Verfahren für Hover Detection

ein Gerät mittels an den Kanten angebrachter Infrarot-Sensoren in die Lage versetzt festzustellen, ob und in welcher Entfernung sich Finger neben dem Gerät befinden. Diese Information kann dann genutzt werden, um mit dem Gerät zu interagieren, zum Beispiel zum Drehen von Bildern. Die Technik eignet sich besonders für kleine Geräte, bei denen bei einer direkten Steuerung mit dem Finger durch Berührung des Bildschirms große Bereiche desselben verdeckt würden.

Mittels einem auf der Rückseite des Geräts angebrachten Touchpad sowie einer auf die Rückseite gerichteten Kamera waren Wigdor u. a. (2007) in der Lage, Interaktion mit dem Finger ohne Verdeckung des Bildschirms auf der Rückseite des Geräts stattfinden zu lassen. Mit dem Touchpad konnte das Gerät gesteuert werden. Damit die Nutzer trotz der Verdeckung der Finger durch des Gerät eine zielgerichtete Berührungen vornehmen konnten, wurde das Bild der Kamera über die auf dem Bildschirm dargestellten Inhalte gelegt, so dass die Fingerposition immer bekannt war. Dieses als *Pseudo-Transparenz* bezeichnete Prinzip wurde von den Nutzern intuitiv verstanden und ermöglichte eine sehr schnelle Nutzung des Geräts.

Eine für kleine und sehr kleine Geräte optimierte Version dieser Technik wurde von Baudisch und Chu (2009) untersucht. Hier wurde wieder die Rückseite des Geräts für die Interaktion verwendet. Die Position der Finger wurde in diesem Prototypen nicht mit einer Kamera erfasst, sondern über die Berührung der rückwärtigen Interaktionsfläche. Das Auslösen einer Aktion geschah durch festeres Drücken auf der Rückseite des Geräts. Die *Pseudo-Transparenz* wurde mangels Kamerabild mit einer computergenerierten Überlagerung des Bildes erzeugt. Die Technik konnte von den Nutzern mit höherer Präzision genutzt werden als die von Vogel und Baudisch (2007) entwickelte und in der Veröffentlichung zum Vergleich genutzte Technik *Shift* (vgl. Abschnitt 2.1.3).

Die Verwendung eines Tiefensensors ermöglichte in der Arbeit von Jones u. a. (2012) die dreidimensionale Nutzung des Raums um das Gerät herum. Sie untersuchten am Beispiel der Navigation in großen Datensets wie Karten oder hochauflösenden Bildern, ob und wie Nutzer von Gesten im Raum zum Verschieben des Sichtfensters und zum Anpassen des Vergrößerungsmaßstabs profitieren konnten. Sie fanden heraus, dass der größere Interaktionsraum dazu genutzt werden kann, eine natürliche, immersive Nutzungserfahrung zu realisieren.

Ebenfalls mit einem Tiefensensor realisierten Kratz u. a. (2012) die Interaktionstechnik *Palm Space*, bei der die Rotation eines dreidimensionalen Objekts mittels Handgesten gesteuert werden kann. In der Evaluation erwies sich ihre Technik als schneller und einfacher als die Verwendung eines virtuellen Trackballs.

2.4 Interaktives kontextsensitives Feedback

Um einem Nutzer durch zusätzliche Informationen Hilfestellung bei der Nutzung einer Anwendung geben zu können, muss ihm diese Information zunächst so vermittelt werden, dass er sie bewusst oder unbewusst wahrnehmen und verarbeiten kann. Im Bereich der mobilen Smartphonenutzung ist dies primär die Darstellung auf dem Bildschirm. Aufgrund der eingeschränkten Bildschirmfläche besteht dafür jedoch nicht unbegrenzt Raum, besonders wenn aus ergonomischen Gründen eine Mindestgröße eingehalten werden soll. Daher werden zusätzlich oft Audiosignale und sogenannte taktile Symbole verwendet, um die Aufmerksamkeit des Nutzers zu erregen und ihn über eingetretene Ereignisse zu unterrichten. Ein solches Ereignis kann auch sein, dass die Hover Detection neue Informationen über einen Finger bereit stellt.

In dieser Arbeit wurde der Schwerpunkt auf die Nutzung visuellen Feedbacks zur Vermittlung der Informationen aus der Hover Detection gelegt, die anderen beiden Kanäle wurden jedoch ebenfalls in kleinerem Rahmen untersucht.

2.4.1 Klassifikation visueller Darstellung von Informationen auf Computern

Information wird in einem typischen Computerprogramm visuell repräsentiert, zum Beispiel durch Texte, Bilder oder Symbole. Grafische Elemente wie Rechtecke und Linien werden verwendet, um Informationen hervorzuheben oder zu gruppieren. Wenn mehr Information dargestellt wird, erhöht sich die Anzahl der visuellen Elemente auf dem Bildschirm. Dies funktioniert allerdings nur so lange, wie freier Platz für die zusätzlichen Elemente zur Verfügung steht.

Eine Möglichkeit dieses Problem zu umgehen ist die automatische Verkleinerung der vorhandenen Elemente, bis genug Platz für die neue Information zur Verfügung steht. Dieses Vorgehen bietet den Vorteil, dass alle Informationen auf einmal dargestellt werden, ohne dass eine zusätzliche Interaktion des Nutzers notwendig ist. Durch die zunehmend kleinere Darstellung ist diese Lösung allerdings limitiert, da die Informationen irgendwann aus technischen (Bildschirmauflösung) oder ergonomischen (Sehvermögen) Gründen nicht mehr problemlos wahrgenommen werden können.

Soll die Größe der Darstellung beibehalten werden, dann kann eine virtuelle Darstellungsfläche verwendet werden, die größer ist als der gerade auf dem Anzeigegerät sichtbare Ausschnitt. Durch Verschieben des Ausschnitts lässt sich dann durch die Informationen navigieren. Alternativ lässt sich die Information strukturiert darstellen, zum Beispiel durch hierarchische Gliederung und mehrstufige Interaktion zur Navigation. Beide Varianten haben den Nachteil, dass der Nutzer mit dem System interagieren muss, um Zugriff auf alle Informationen zu erhalten.

Plaisant, Carr und Shneiderman (1995) entwickelten eine Taxonomie zu Gliederung der Darstellungsarten von Informationen und prägten dabei einige Bezeichnungen wie „Übersicht und Detail“ oder „Fisheye-Ansicht“, die von anderen Autoren aufgegriffen wurden. Cockburn, Karlson und Bederson (2009) bieten eine Übersicht, wie diese Mechanismen einzeln oder kombiniert verwendet werden können, um Information darzustellen. Sie unterscheiden vier Kategorien:

- Vergrößerung der Darstellung mit zeitlicher Trennung zwischen Übersicht und Detailansicht
- Vergrößerung der Darstellung mit räumlicher Trennung zwischen den beiden Ansichten
- Übersicht mit eingebetteter Detailansicht
- Hinweisorientierte Hervorhebung von Informationen

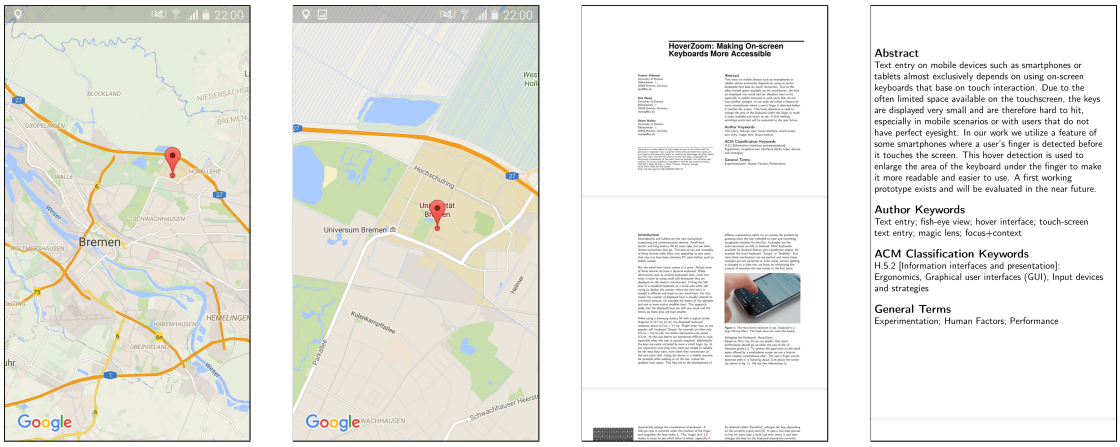
Sie werden im Folgenden kurz vorgestellt.

Vergrößerung mit zeitlicher Trennung

Eine weit verbreitete Lösung für das Problem der kleiner werdenden Darstellung bei steigender Informationsdichte ist das gezielte Vergrößern einer Auswahl der dargestellten Informationen. Während der Nutzer mit der vergrößerten Darstellung interagiert, hat er zunächst keine Möglichkeit, auf die nicht dargestellten Informationen zuzugreifen. Falls dies notwendig sein sollte, kann er – je nach Implementierung – entweder die in der Vergrößerung dargestellte Auswahl direkt verändern oder zum Ursprungsmaßstab zurückkehren, eine neue Auswahl treffen und diese wieder vergrößern. Die vergrößerte Darstellung kann als eine Art Fenster in die virtuelle Darstellung begriffen werden.

Beispiele für diese Vorgehensweise sind Kartendarstellungen, bei der der Vergrößerungsmaßstab vom Nutzer frei gewählt werden kann (vgl. Abb. 2.15a und 2.15b) oder Vergrößerungen von Dokumenten oder Webseiten auf Smartphones, bei denen eine ansonsten zu kleine Textdarstellung oft durch eine gezielte Vergrößerung auf die Spaltenbreite des Textes behoben werden kann (Abb. 2.15c und 2.15d). Beim Lesen des Textes kann der Nutzer den Text dann so verschieben, dass die ungelesenen Zeilen in den Ausschnitt geschoben werden.

Vorteil dieser Darstellung ist der vom Nutzer meist frei wählbare Vergrößerungsmaßstab. Nachteilig ist, dass beim Wechsel in die vergrößerte Ansicht der Kontext der Information verloren geht. Der Nutzer muss also ein abstraktes Modell der Umgebung der aktuell vergrößerten Informationen im Gedächtnis behalten, um die Ansicht effektiv nutzen zu können. Bederson und Boltman (1999) haben allerdings festgestellt, dass der Einsatz von Animationen beim Vergrößern bzw. Verkleinern des Ausschnitts dem Nutzer helfen kann, den Kontext zu behalten. Außerdem steigt mit höherer Vergrößerung



(a) Kartenübersicht (b) Kartendetail (c) Textübersicht (d) Textdetail

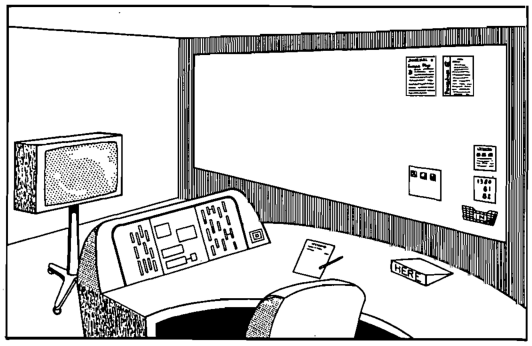
Abbildung 2.15: Verschiedene Anwendungen von Übersicht und Detailansicht mit temporaler Trennung. Die Ansichten werden durch Interaktion des Nutzers geändert und können nur nacheinander angezeigt werden. (Karten: Google)

der Aufwand zum Verschieben des Bildausschnitts zu einer bestimmten Stelle, da weniger Information auf einem gleichgroßen Ausschnitt dargestellt wird. Eine Strategie zum Lösen dieses Problems ist, zunächst die Vergrößerung so weit zu verringern, dass das Ziel mit wenig Aufwand erreicht werden kann, um dort die Vergrößerung wieder zu erhöhen (Baudisch u. a. 2002). Eine intuitive Steuerung des Vergrößerungsgrades stellten Harrison und Dey (2008) vor, indem sie die Dichte der dargestellten Informationen an die Entfernung des Nutzers vom Bildschirm koppelten. Ein Nach-vorne-Lehnen des Nutzers erhöhte dabei die Informationsdichte, während ein Zurück-Lehnen ihn verringerte.

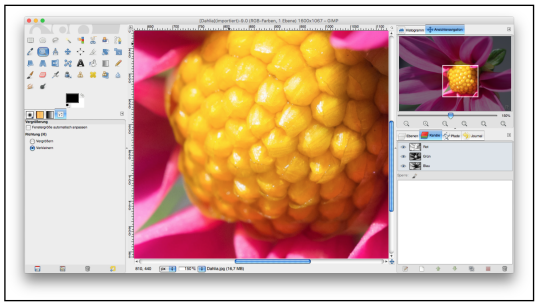
Vergrößerung mit räumlicher Trennung

Eine weitere Möglichkeit bei der Verwendung einer vergrößerten Darstellung besteht darin, nicht die komplette Ansichtsfläche für die Vergrößerung zu verwenden. Die Detailansicht wird räumlich getrennt von der Übersicht angezeigt, zum Beispiel wie in Abbildung 2.16b seitlich versetzt. Eine erste Beschreibung eines solches System stammt von Spence und Apperley (1982). Sie sprechen von einem *bifocal display* (dt. etwa Anzeige mit zwei Fokuspunkten), das ein kontextwahrendes Durchsuchen großer Informationsmengen erlauben soll. Neben einer Übersicht der gesamten Daten erlaubt eine zentrale Detailansicht die komplette Darstellung vom Nutzer ausgewählter Artikel. Bei Cockburn, Karlson und Bederson (2009) wird eine solche Anordnung als *Overview+Detail* (dt. Übersicht und Detailansicht) bezeichnet.

Ein Vorteil dieser Darstellung ist, dass die Übersicht im mentalen Modell des Nutzers erhalten bleibt. Ungünstig ist jedoch, dass dieser zum Betrachten der Vergrößerung den



(a) Schreibtisch mit *bifocal display*. (Abb. aus Spence und Apperley 1982)



(b) Detail- und Übersicht im Bildbearbeitungsprogramm *Gimp*.

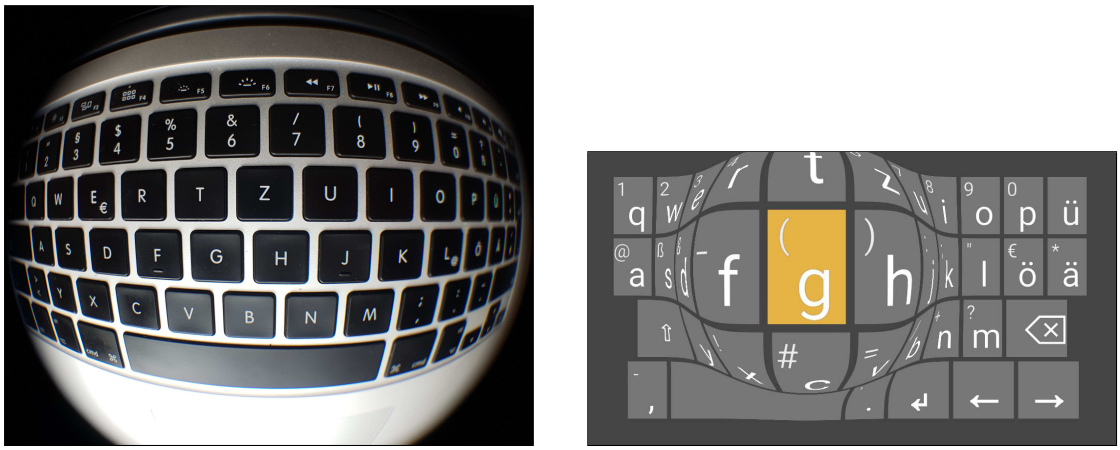
Abbildung 2.16: Beispiele für *Overview+Detail*-Darstellungen.

Fokus seiner Aufmerksamkeit verschieben muss, was negative Auswirkungen auf die Nutzbarkeit oder das Nutzungserlebnis haben kann.

Übersicht mit eingebetteter Detailansicht

Zur Vermeidung der temporalen oder spatialen Trennung zwischen Kontext und Detailansicht kann eine kombinierte Darstellung gewählt werden. Bei dieser wird die Detailansicht in die Darstellung des Kontextes eingebettet. Die Vergrößerung ist räumlich begrenzt und nimmt nicht die komplette Ansicht ein. Cockburn, Karlson und Bederson (2009) bezeichnen diese Art der eingebetteten Darstellung als *Focus+Context* (dt. Fokus und Kontext).

Eine bestimmte Ausprägung davon ist die Fisheye-Darstellung, die in Anlehnung an die gleichnamige Abbildung aus der Fotografie erfolgt (vgl. Abb. 2.17a und 2.17b). Sie bekommt ihren Namen daher, dass die Projektion durch eine Fisheye-Linse ein Bild ergibt, das der von einem Fischauge erzeugten Verzerrung ähnlich ist und das Motiv mit einer variablen Vergrößerung abbildet. Solch eine Darstellung bietet den Vorteil, dass der Fokus im Kontext eingebettet dargestellt werden kann. Allerdings muss für die Darstellung der Vergrößerung im Zentrum des Fisheye erst Raum geschaffen werden. Dazu können entweder benachbarte Informationen verkleinert und verzerrt dargestellt werden, oder die zur Darstellung der Informationen notwendige Gesamtfläche vergrößert werden, indem benachbarte Elemente verschoben werden um Platz für die Vergrößerung zu schaffen. Dies führt insbesondere dann zu Problemen, wenn der Nutzer den Fokus und damit das Fisheye auf ein neues Ziel bewegen will. Bedingt durch die Vergrößerung bewegt sich beim Verschieben des Fisheye die darin dargestellte Information schneller als das Fisheye an sich (Gutwin 2002). Zudem bewegen sich beim Verschieben auch die Trefferzonen darin enthaltener interaktiver Elemente, sodass der Nutzer nun kein statisches Ziel mehr treffen muss, sondern ein dynamisches (siehe



(a) Tastatur mit Fisheye-Linse fotografiert. Erkennbar ist die typische runde Verzerrung paralleler Linien.

(b) Bildschirmtastatur mit Fisheye-ähnlicher Vergrößerung. Auch hier ist die runde Verzerrung sichtbar.

Abbildung 2.17: *Focus+Context* am Beispiel von Fisheye-Abbildungen.

Abb. 2.18). Dieser Effekt kann beispielsweise in der Programmanzeige von Mac OS X (dem „Dock“) beobachtet werden (vgl. Abb. 2.18). Dies kann das gezielte Interagieren mit dem Fisheye schwieriger machen.

Fisheye-Darstellungen wurden auch schon für Bildschirmtastaturen verwendet. Raynal und Truillet (2007) implementierten eine Tastatur, die auf einem PDA die Tasten in der Nähe der aktuellen Stylus-Position mit einer Fisheye-ähnlichen Vergrößerung darstellte. Obwohl das namensgebende Fisheye-Objektiv immer einen runden Verzerrungseffekt erzeugt, ist die Technik am Computer weniger eingeschränkt, so dass auch rechtwinklige Vergrößerungen umgesetzt werden können (Rauschenbach, Jeschke und Schumann 2001). Aufgrund der ebenfalls rechtwinkligen Struktur einer typischen Tastaturdarstellung eignet sich diese Technik möglicherweise besonders gut für die Darstellung einer Tastatur.

Zur Abgrenzung sei hier noch die generalisierte Definition einer Fisheye-Darstellung aufgeführt. Sie wurde von Furnas (1986) aufgestellt und beschreibt, wie der Detailgrad an dargestellter Information in Abhängigkeit der Distanz zwischen dem Fokus des Anwenders und der jeweils dargestellten Information angepasst werden kann. Ist diese Distanz größer als der *Degree of Interest* für den Nutzer, so wird die Information ausgeblendet, der dargestellte Detailgrad nimmt mit steigender Distanz zwischen Fokus des Anwenders und Lage der dargestellten Information also ab. Ein Beispiel für eine solche generalisierte Fisheye-Darstellung ist das automatische Ein- oder Ausblenden von Funktionen einer Programmierumgebung in Abhängigkeit von der gerade zu erledigenden Aufgabe, wie es zum Beispiel die aufgabenorientierte Erweiterung *Mylyn* in der IDE *Eclipse* anbietet (Kersten und Murphy 2005). Diese Definition sei hier allerdings nur zur Abgrenzung erwähnt, bei der folgenden Untersuchung

der Darstellung kontextsensitiver Hover-Informationen wurde sie nicht weiter untersucht.

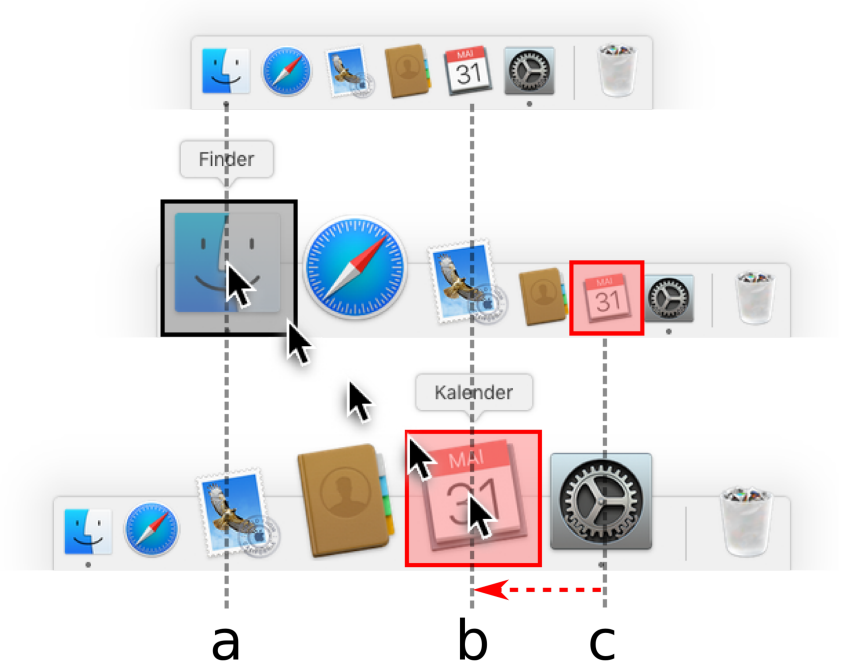
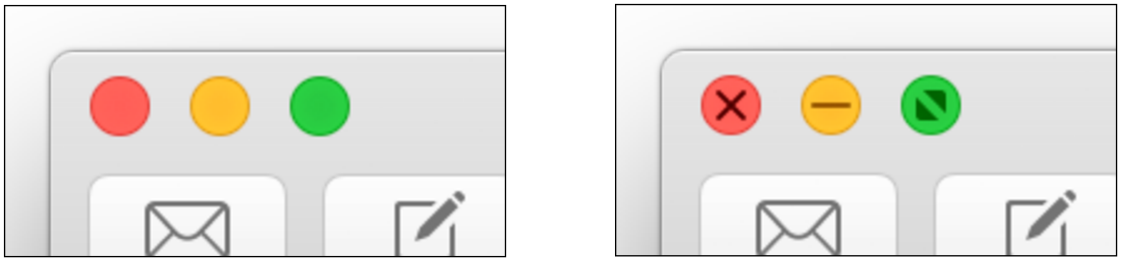


Abbildung 2.18: Fisheye-Darstellung in Mac OS X.
Oben: Das Dock im Normalzustand. Trefferzone für das *Finder*-Symbol bei *a*, für den Kalender bei *b*.
Mitte: Mauszeiger über dem *Finder*-Symbol. Symbole direkt unter und neben dem Mauszeiger werden vergrößert, um ein Anklicken zu erleichtern. Um Raum für die Vergrößerung zu schaffen, bewegen sich weiter entfernte Symbole nach außen. Die Trefferzone für den Kalender scheint nun bei *c* zu sein.
Unten: Um das Kalender-Icon zu treffen, muss der Mauszeiger aber tatsächlich nur bis *b* bewegt werden, da sich die Vergrößerung der Symbole mit dem Mauszeiger verschiebt. Das vom Nutzer angepeilte Ziel bewegt sich dabei wie durch den roten Pfeil dargestellt gegenläufig zur Mausbewegung. (Abbildung nach Cockburn, Karlson und Bederson 2009)

Hinweisorientierte Darstellung

Eine weitere Möglichkeit die dargestellte Informationsdichte zu erhöhen besteht darin, Elemente im Fokus des Anwenders anders darzustellen als außerhalb. Typische Beispiele sind die farbige Hervorhebung von interaktiven Steuerelementen, wenn sie mit



(a) Schaltflächen im Normalzustand (b) Beim Überfahren mit dem Mauszeiger

Abbildung 2.19: Hinweisorientierte Hervorhebung der Fensterkontrollelementen in Mac OS X 10.11. Die eingeblendeten Symbole zeigen dem Nutzer, welcher Kreis welche Aktion hervorrufen wird.

dem Mauszeiger berührt werden (vgl. Abschnitt 2.3.1). Ein weiteres Beispiel sind die in Mac OS X verwendeten Schaltflächen zur Kontrolle des aktuellen Fensters, die bei Kontakt mit dem Mauszeiger ein zusätzliches Symbol einblenden (vgl. Abb. 2.19).

3 Einsatz und Bewertung von Hover Detection auf Mobilgeräten

Texteingabe auf Smartphones ist eine Kompromisslösung, bei der bei Verwendung großer Tasten zwischen wachsendem Platzbedarf auf der einen Seite und steigender Nutzbarkeit auf der anderen Seite abgewogen werden muss. Je nach Bedürfnissen und Fähigkeiten des Nutzers kann dieser Kompromiss für ihn mehr oder weniger gut geeignet sein. Vor allem bei eingeschränkter Sehfähigkeit führt eine zu kleine Darstellung zu Nachteilen bei der Nutzung. Durch die Einbindung zusätzlicher Sensoren wie der Hover Detection kann die Darstellung jedoch kontextsensitiv angepasst und vergrößert werden. Dies könnte die Nutzerfreundlichkeit für bestimmte Zielgruppen wie ältere Nutzer oder solche mit eingeschränkten Sehfähigkeiten deutlich verbessern.

Die Untersuchung von Hover Detection bei Texteingabe auf mobilen Geräten hat sehr viele querschneidende Verbindungen zu anderen Forschungsgebieten. Da in dieser Arbeit die Eignung von Hover Detection zur Verbesserung von Texteingabe auf Smartphones untersucht werden soll, wird der Schwerpunkt auf der Verbesserung der Eingabe an sich liegen und nicht auf nachgelagerten Korrekturmechanismen wie einer Rechtschreibkorrektur oder Vorhersagemechanismen.

Für die Untersuchung im Rahmen dieser Dissertation wurde der Fokus auf eine experimentelle Untersuchung anhand von Prototypen gelegt. Diese wurden für aktuell auf dem Markt verfügbare Geräte entwickelt und mit geeigneten Versuchspersonen getestet werden.

3.1 Geplantes Vorgehen

Zunächst werden die intrinsischen Eigenschaften (Latenz, Genauigkeit) der Hover Detection auf den verwendeten Smartphones untersucht, um die Ergebnisse der Experimente zur Usability besser einordnen zu können. Parallel werden sowohl die prinzipielle Eignung von Hover Detection zur Verbesserung der Zielauswahl als auch die konkrete Anwendung zur kontextbezogenen Vergrößerung in einer Bildschirmtastatur untersucht. In mehreren iterativen Experimenten wird auf die vorherigen Ergebnisse aufgebaut, um zusammen mit Mitgliedern verschiedener Zielgruppen eine für sie möglichst gute Lösung zu finden.

3.2 Technische und physikalische Eigenschaften der verwendeten Geräte

Für die folgenden Experimente wurden zwei verschiedene Smartphones verwendet, die zum Zeitpunkt der Untersuchungen auf dem Markt erhältlich waren und Hover Detection unterstützen: Das Samsung Galaxy S4 und Samsung Galaxy S5. Beide waren zum Zeitpunkt ihres Erscheinens die Top-Modelle des Herstellers und wurden mit dem Betriebssystem *Android* betrieben. Dieses war daher die natürliche Wahl für die Implementierung der in den folgenden Experimenten verwendeten Prototypen.

3.2.1 Das mobile Betriebssystem Android

Android wird von der Firma Google Inc. entwickelt und ist speziell auf die Fähigkeiten und Anforderungen von Mobilgeräten hin angepasst. Es ist sehr modular aufgebaut und unterstützt neben Berührungssteuerung auch die klassische Bedienung über Tastatur und Maus, sofern solche an das Gerät angeschlossen werden. Google veröffentlicht die Quelltexte für das Betriebssystem, sodass andere Hersteller angepasste Versionen von Android mit eigenen, zusätzlichen Features implementieren können. Samsung hat als eines dieser zusätzlichen Features Hover Detection implementiert und dafür die Unterstützung der Mauseingabe von Android erweitert. Normalerweise löst nur das Bewegen eines Mauszeigers über ein Element der grafischen Nutzerschnittstelle ein sogenanntes *Hoverevent* aus. Dieser Mechanismus wurde erweitert, so dass die als *Air-View* bezeichnete Hover Detection ebenfalls diese Events auslöst. Neue Versionen des Betriebssystems erweitern den Umfang seiner Fähigkeiten. Solche neuen Fähigkeiten werden über Programmierschnittstellen verfügbar gemacht, die bei Android in verschiedenen *Leveln* zur Verfügung stehen. Der Event-Typ für Hover wurde in Android erst mit Android 4.0 und dem damit verbundenen API-Level 14 eingeführt, daher ist diese Android-Version die Mindestvoraussetzung auf dem Gerät. Dies stellt in der Praxis jedoch keine Einschränkung dar, da die entsprechenden Geräte von Samsung mindestens mit der Android 4.2 auf den Markt kamen.

Um die relevanten Parameter der Geräte bei der Beschreibung der folgenden Experimente nicht stets wiederholen zu müssen, werden sie einmal hier beschrieben und dann bei Bedarf referenziert.

3.2.2 Samsung Galaxy S4

Das Galaxy S4 wurde am 14. März 2013 angekündigt und war ab dem 27. April 2013 mit der Typenbezeichnung *GT-I9500* im Handel verfügbar.

Ursprünglich war eine von Samsung angepasste Version des Betriebsssystems Android in der Version 4.2.2 auf dem Gerät vorinstalliert. Nach mehreren Updates ist aktuell und

Abbildung 3.1: Samsung Galaxy S4 (links) und Galaxy S5 (rechts).

als voraussichtlich letztes Update Version 5.0.1 von Android für das Gerät verfügbar. Als Prozessor dient ein in dem *System on a chip* (*SoC*) *Qualcomm Snapdragon 600* enthaltener Vierkern-Prozessor mit ARM-Befehlssatz, der mit einer Taktrate von 1,9 GHz rechnet. Ebenfalls auf dem SoC verbaut sind der Grafikchip *Adreno 320* für hardwarebeschleunigte Darstellung und 2 GiB Arbeitsspeicher. Das Gerät ist mit einem AMOLED-Bildschirm im Seitenverhältnis 9:16 ausgestattet, der bei einer Diagonalen von 12,7 cm (5 Zoll) 1080 × 1920 Bildpunkte auflöst. Umgerechnet ergibt sich dadurch eine Auflösung von 441 ppi (*pixel per inch*, Pixel pro Zoll). Das Gerät hat eine virtuelle Auflösung von 360 × 640 dp[1]. Mittels des kapazitativen Sensors (vgl. 2.3.2) im Bildschirm können sowohl mehrere gleichzeitige Berührungen als auch ein einzelner, über dem Bildschirm schwebender Finger erkannt werden. Die dargestelltem Bildschirminhalte werden mit einer Rate von 60 Hz aktualisiert, was seit Android 4.1 Standard ist (Thoma 2012).

Unter der Bezeichnung *Galaxy S4* wurden mehrere Hardware-Generationen verkauft, die anhand der Typenbezeichnungen *GT-I9505*, *GT-I9515* und *GT-I9506* unterschieden werden können. Je nach Version verfügt das Gerät über einen leicht schnelleren Prozessor oder zusätzlich die Möglichkeit, für den Mobilfunk das LTE-Band zu nutzen. Am Aussehen, den Sensoren oder dem Bildschirm ändern diese Varianten jedoch nichts.

Für die folgenden Untersuchungen standen die Varianten *GT-I9505* und *GT-I9515* zur Verfügung.

3.2.3 Samsung Galaxy S5

Das Galaxy S5 wurde als Nachfolger des Galaxy S4 am 24. Februar 2014 angekündigt und war ab dem 11. April 2014 im Handel verfügbar.

Es erschien mit einer von Samsung angepassten Android-Version auf Basis von Android 4.4.2. Das Gerät erhielt zwei Mal Updates auf die jeweils neueste Android-Version. So erschien zunächst Android 5.0.1, inzwischen ist als voraussichtlich letzte Version auch Android 6.0.1 verfügbar. Das Gerät ist in seiner Hardware-Ausstattung dem Vorgänger relativ ähnlich. Der AMOLED-Bildschirm löst weiterhin mit 1080 × 1920 Bildpunkten auf, hat aber eine Diagonale von 12,95 cm (5,1 Zoll) und damit umgerechnet 432 ppi. Es besitzt ebenfalls eine virtuelle Auflösung von 360 × 640 dp. Die Fähigkeiten bezüglich Berührung und Hover Detection sind gleich derer des Galaxy S4. Als *SoC* ist nun ein *Qualcomm Snapdragon 801* verbaut, dessen Vierkern-Prozessor mit 2,5 GHz taktet. Der Grafikprozessor ist ein *Adreno 330* und es stehen weiterhin 2 GiB Arbeitsspeicher zur Verfügung.

[1] Aufgrund der sehr fragmentierten Android-Landschaft werden von Programmierern angegebene Größen in sog. *density-independent pixels* (*dp*, übersetzt etwa „auflösungsunabhängige Bildpunkte“) umgerechnet (Google Inc. 2016). Damit wird erreicht, dass bei der Gestaltung von Oberflächen die tatsächliche Auflösung des Endgeräts nicht bekannt sein muss. Das Galaxy S4 fällt in die Kategorie *xxhdpi*, in welcher ein *density-indepentent pixel* neun (3 × 3) physikalischen Bildpunkten entspricht.

3.2.4 Nachfolgegenerationen

Ab dem Galaxy S6, welches als Nachfolger des S5 am 10. April 2015 vorgestellt wurde, ist keine Hover Detection mehr in die Geräte der Galaxy-S-Baureihe integriert. Die mit einem größeren Bildschirm ausgestattete Baureihe Galaxy Note unterstützt das Feature noch, allerdings nur bei Verwendung des mitgelieferten Stiftes.

Damit ist aktuell (Mai 2017) kein Smartphone mehr auf dem Markt, das über dem Bildschirm schwebende Finger erkennen kann.

3.3 Untersuchung der Hover Detection auf dem Samsung Galaxy S4 und S5

Die folgenden Experimente dienten zur Untersuchung der spezifischen Eigenschaften und Limitierungen der als *AirView* bezeichneten Hover Detection auf den Smartphones Samsung Galaxy S4 und Galaxy S5.

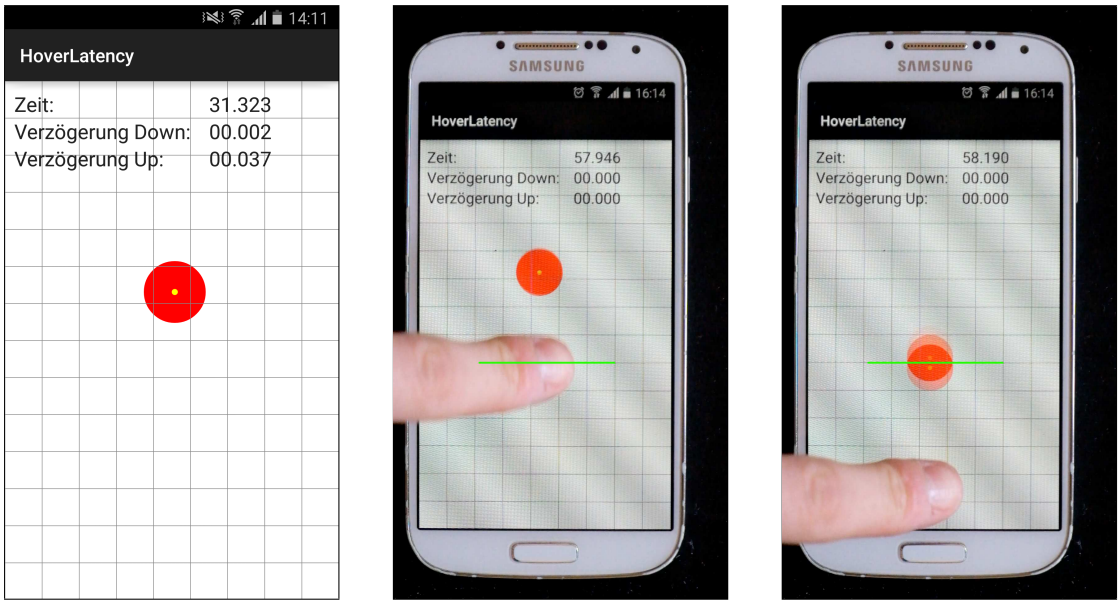
3.3.1 Latenz der Hover Detection

Bereits bei anfänglichen Tests mit der Hover Detection fiel auf, dass die gemeldete Position der tatsächlichen Position des Fingers über dem Bildschirm merklich „hinterher zog“, also eine Latenz zwischen Bewegung des Fingers und Detektion im Gerät existierte. Solche Latenzen können einen großen Einfluss auf die Nutzbarkeit eines Systems haben, werden von den Geräteherstellern jedoch nicht veröffentlicht.

Methodik

Zur Messung der Latenz wurde eine Anwendung entwickelt, welche die aktuelle erfasste Position des Fingers sowohl während des Schwebens als auch beim Berühren des Bildschirms über einem Gittermuster darstellte. Mit einer Videoanalyse der dargestellten Informationen konnte so die Latenz ermittelt werden.

Teilnehmer Das Experiment wurde nur von einer Person (dem Autor) durchgeführt, da hier die technischen Eigenschaften von Geräten untersucht wurden. Es ist davon auszugehen, dass die Ergebnisse bei Durchführung durch andere Personen und für andere, baugleiche Geräte ebenso gelten werden.



(a) Der rote Punkte stellt die aktuelle Hoverposition des Fingers dar

(b) Einzelbild aus der Videoanalyse mit Markierung der Fingerposition

(c) Mit 244 ms Verzögerung folgt die Hoverposition nach

Abbildung 3.2: Experiment zur Bestimmung der Latenz der Hover Detection. Das in b) und c) sichtbare diagonale Gradientenmuster ist ein Aliasing-Artefakt aus der Videoaufnahme.

Apparat Als Versuchsapparat diente eine Anwendung auf einem Samsung Galaxy S4 (vgl. 3.2.2, im folgenden *GS4*) und einem Samsung Galaxy S5 (vgl. 3.2.3, im folgenden *GS5*), die beide mit Android 5.0.1 betrieben wurden und Hover Detection aktiviert hatten.

Die Anwendung bestand aus einem Startbildschirm, auf dem der Typ des Experiments und einige Optionen zur Datenerfassung ausgewählt werden konnten. Danach erschien die grafische Oberfläche des eigentlichen Experiments wie in Abbildung 3.2a dargestellt. Sie bestand aus einer weißen Fläche auf der die aktuelle Position des Fingers grafisch markiert wurde. Falls vom Gerät ein Finger im Schwebezustand erkannt wurde, zeichnete die Anwendung einen roten Kreis mit 200 Pixeln Durchmesser (entspricht 11,52 mm auf dem GS4, 11,76 mm auf dem GS5) mit dem Mittelpunkt auf der erfassten Position. Berührte der Finger das Display, dann wurde die Kreismarkierung in blauer Farbe dargestellt. Um die spätere Auswertung zu vereinfachen wurde jeweils zusätzlich noch ein kleiner gelber Punkt mit 20 Pixeln Durchmesser (entspricht 1,15 mm auf dem GS4, 1,18 mm auf dem GS5) in den Mittelpunkt der großen farbigen Kreise gezeichnet.

Als Hilfestellung für die spätere Videoanalyse wurde ein fortlaufender Millisekundenzähler dargestellt sowie ein quadratisches Gittermuster mit Linienabstand von 100 Pixeln

(das entspricht einer Länge von 5,76 mm auf dem SGS4 und 5,88 mm auf dem GS5) über der gesamten Fläche der Anwendung gezeichnet. Immer wenn der Finger den Bildschirm berührte, wurde die zeitliche Differenz zwischen der letzten Aktualisierung der Hoverposition und dem Zeitpunkt der Bildschirmberührung berechnet und im oberen Bereich der Anwendung in Millisekunden angezeigt. Nach Anheben des Fingers wurde die zeitliche Differenz zwischen der letzten Berührung und dem ersten direkt folgenden Hoverevent berechnet und ebenfalls im oberen Bereich der Anwendung angezeigt.

Zwei Beispiele aus der Videoanalyse (Abb. 3.2b und 3.2c) zeigen, wie anhand dieser Daten die Latenz gemessen werden kann.

Vorgehensweise Zur Messung der Verzögerung wurde eine Kamera auf ein Stativ montiert auf einem Tisch platziert. Sie nahm mit 60 Einzelbildern pro Sekunde und einer Auflösung von 1920×1080 Pixeln das darunter liegende Smartphone auf.

Auf dem Smartphone wurde die oben beschriebene Anwendung zur Latenzmessung gestartet. Dann wurde in geringer Höhe ein einzelner Finger mehrmals über dem Bildschirm vor und zurück bewegt, so dass die Hover Detection ihn erkennen konnte, sichtbar an der roten Markierung.

Darauf folgend wurde der Finger einige Male direkt auf dem Bildschirm vor und zurück bewegt, um auch die Verzögerung des Touchscreens messen zu können.

Zuletzt wurde der Bildschirm mehrmals angetippt, um die Verzögerungen zwischen Hover Detection, Bildschirmberührung und wieder Hover Detection messen zu können.

Nach Abschluss der Messung wurde die Videoaufnahme für die weitere Analyse von der Kamera auf einen PC übertragen. Für die Messung der Schwebe- und der Berührungslatenz wurde der auf dem Bildschirm des Smartphones dargestellte Zeitstempel t_f beim Überstreichen des Fingers einer Gitterlinie festgehalten. Danach wurde im Einzelbildvorlauf der Zeitpunkt t_m gesucht, bei dem die auf dem Bildschirm dargestellte Markierung der Schwebe- oder Berührungsposition die selbe Gitterlinie überschritt. Aus der Differenz des Zeitstempels zu diesem Zeitpunkt und dem vorher festgehaltenen Zeitstempel lässt sich die Latenz l berechnen:

$$t_m - t_f = l \tag{3.1}$$

Für die Analyse der Verzögerungen zwischen Hover- und Touchevents wurde das Video immer dann gestoppt, wenn der Finger beim Antippen den Bildschirm berührt hatte und wieder angehoben wurde, erkennbar am Farbwechsel der Positionsmarkierung von rot nach blau und wieder nach rot. Die gemessenen und angezeigten Werte für die Verzögerung beim Absenken und Anheben des Fingers wurden ebenfalls festgehalten.

Gerät	Fingerbewegung	Mittelwert
Galaxy S4	Schweben	232 ms ± 30 ms
	Berühren	99 ms ± 10 ms
	Absenken	7 ms ± 5 ms
	Anheben	38 ms ± 2 ms
Galaxy S5	Schweben	256 ms ± 15 ms
	Berühren	98 ms ± 6 ms
	Absenken	2 ms ± 2 ms
	Anheben	49 ms ± 1 ms

Tabelle 3.1: Ergebnisse der Messung der Latenz der Hover Detection

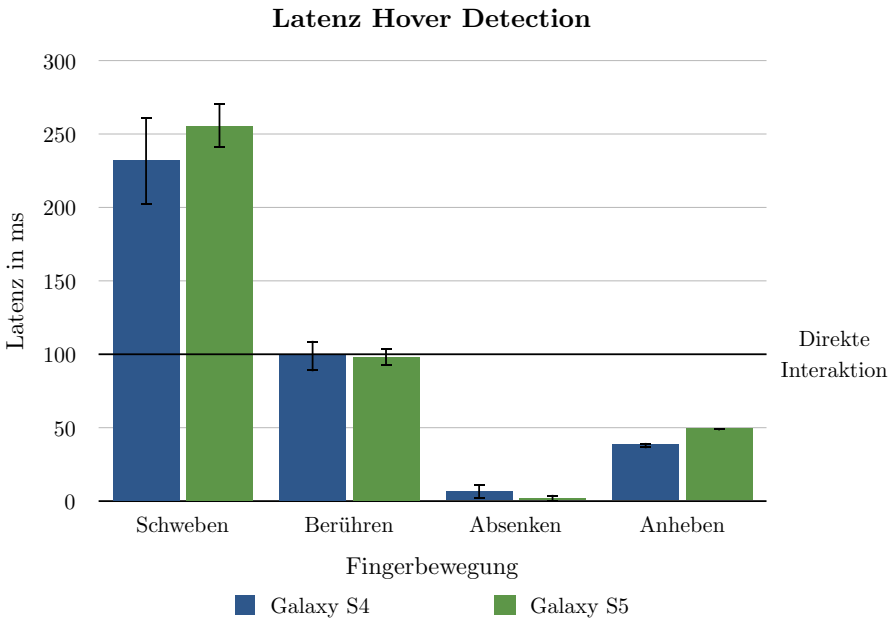


Abbildung 3.3: Ergebnisse der Messung der Latenz der Hover Detection

Auswertung Insgesamt wurden auf jedem Gerät jeweils 25 Werte für die Verzögerung im Schwebemodus und bei Berührung des Bildschirms erfasst. Außerdem wurden 16 Berührungen des Bildschirms bezüglich der zeitlichen Verzögerung zwischen Hover- und Touchevents beim Absenken und beim Anheben des Fingers untersucht.

Die Mittelwerte und Standardabweichungen der Messungen sind in Diagramm 3.3 und Tabelle 3.1 dargestellt.

Diskussion der Ergebnisse Anhand der gemessenen Latenz der Hover Detection wird schnell klar, dass der Einsatz in interaktiven Szenarien zumindest problembehaftet sein wird. Der Mittelwert der Latenz der Hover Detection liegt beim Galaxy S4 bei 232 ms

und beim Galaxy S5 bei 256 mm, dies jedoch relativ konstant bei einer Standardabweichung von 30 ms bzw. 15 ms. In beiden Fällen ist liegt die Latenz deutlich über der von Miller (1968) empfohlenen Schwelle von 100 ms, unterhalb derer Nutzer die Interaktion mit einem System als unmittelbar empfinden. Nach MacKenzie und Ware (1993) kann davon ausgegangen werden, dass Nutzer bei der Nutzung eines System mit einer so hohen Latenz deutlich schlechtere Zeiten bei der Berührung eines Ziels und höhere Fehlerraten erzielen werden.

Selbst bei direkter Berührung des Bildschirms wird diese Grenze nur sehr knapp eingehalten, beide Geräte haben eine ungefähre Verzögerung von 100 ms bei der Darstellung von Reaktionen auf Touchevents. Maximal 16,7 ms dieser Latenz lassen sich bei einer Bildwiederholrate von 60 Hz damit erklären, dass die Reaktion auf ein Hover- oder Touchevent erst bei der nächsten Aktualisierung des Bildschirms ausgegeben werden kann. Auch die Zeit für die Berechnung der interaktiven Darstellung im verwendeten Prototypen kann nicht als Erklärung für die Latenz dienen, da weder Prozessor noch Grafikchip laut Anzeige im Android-Debugger annähernd ausgelastet wurden.

Die Werte für die Verzögerung zwischen Hover- und Touchevents beim Absenken mit durchschnittlich 7 ms (GS4) bzw. 2 ms (GS5) und beim Anheben des Fingers mit durchschnittlich 38 ms (GS4) bzw. 49 ms (GS5) belegen, dass auch das Eventsystem von Android schnell genug ist, um Events mit niedriger Latenz zu verteilen und hier als Verursacher ausscheidet.

Eine mögliche Erklärung für die beobachtete Latenz der Hover Detection wäre eine interne Interpolation gemessener Schwebepositionen über mehrere aufeinander folgende Messungen hinweg. Aus Sicht eines Softwareentwicklers erscheint diese Erklärung vor allem dann sinnvoll, wenn die Rohdaten aufeinanderfolgender Messwerte eine hohe Streuung aufweisen und ansonsten nicht sinnvoll vom Programmierer einer App verwendet werden könnten. Es ist allerdings zu hinterfragen, ob beim Anwenden dieser hypothetischen Interpolation zum Erreichen eines besseren Nutzererlebnisses nicht über das Ziel heraus geschossen wurde. Nach Teather u. a. (2009) hat die Latenz eines interaktiven Systems einen deutlich stärkeren Effekt auf die Nutzer als ein leichtes Zittern (engl. *jitter*) der gelieferten Koordinaten, so dass am Ende zwar möglicherweise ein besserer Ersteindruck beim Nutzer entsteht, die Nutzbarkeit an sich dadurch jedoch mehr beeinträchtigt wird als technisch notwendig.

3.3.2 Genauigkeit der Hover Detection

Wie in Abschnitt 2.3.2 beschrieben, werden für die Feststellung der Hoverposition und der Touchposition eines Fingers über bzw. auf dem Bildschirm verschiedene sensorische Verfahren verwendet. Wenn Hover Detection für die kontextsensitive Einblendung von visuellem Feedback verwendet werden soll, um das Berühren von Elementen

auf dem Bildschirm zu erleichtern, müssen die Hover- und Schwebeposition übereinstimmen. Wenn nicht, würde das visuelle Feedback an der falschen Stelle eingeblendet werden. Nachdem aus Experiment 3.3.1 bereits bekannt ist, dass die Hover Detection nur mit merklicher Verzögerung funktioniert, sollte zusätzlich untersucht werden, wie genau die Hover Detection ist und welchen Einfluss die Bewegungsgeschwindigkeit des Fingers in der Praxis auf die Genauigkeit der Hover Detection hat.

Methodik

Zur experimentellen Untersuchung dieser Fragen wurde eine einfache Anwendung entwickelt und mit einigen Versuchspersonen getestet.

Teilnehmer Das Experiment wurde mit acht Teilnehmern im Alter von 27 bis 33 Jahren durchgeführt, die dem Versuchsleiter persönlich bekannt waren. Alle Versuchspersonen besaßen ein Smartphone und nutzten es regelmäßig.

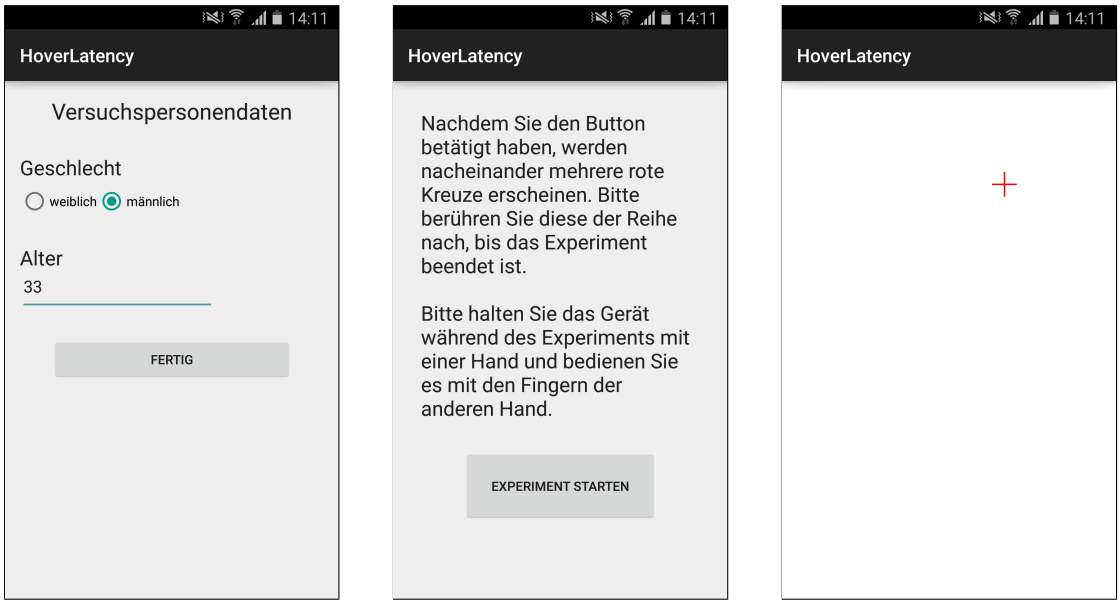
Apparat Das Experiment war ebenfalls Bestandteil der Anwendung, mit der auch die Latenz der Hover Detection gemessen wurde (vgl. 3.3.1). Getestet wurde mit einem Samsung Galaxy S4 (vgl. 3.2.2) und einem Samsung Galaxy S5 (vgl. 3.2.3), beide mit Android 5.0.1.

Nach Betätigen einer Schaltfläche wurden zunächst demografische Daten des Versuchsteilnehmers abgefragt (Abb. 3.4a) und in eine Log-Datei geschrieben. Anschließend zeigte die Anwendung Instruktionen für das Experiment an (Abb. 3.4b). Über eine Schaltfläche konnte der Nutzer das Experiment nach Lesen der Anweisungen selbst starten.

Die Aufgabe bestand daraus, nacheinander vom System zufällig bestimmte und mit einem roten Kreuz auf dem Bildschirm markierte Koordinaten zu berühren (Abb. 3.4c). Eine Berührung des Bildschirms wurde durch kurzes Einblenden eines blauen Punktes an der Kontaktposition bestätigt. Dann wurde die aktuelle Zielmarkierung ausgeblendet und die nächste angezeigt. Beim Absenken des Fingers zum Berühren des Bildschirms erfasste das System kontinuierlich die von der Hover Detection gemeldeten Koordinaten des Fingers.

Bei der Berührung des Bildschirms wurden folgende Daten in eine Log-Datei geschrieben:

- die Zielkoordinaten
- der Zeitpunkt des Einblendens der Zielkoordinaten
- die tatsächlich berührten Koordinaten
- der Zeitpunkt der Berührung



(a) Abfrage der Teilnehmer-Daten

(b) Anzeige der Instruktionen

(c) Anzeige des Zielpunkts der Berührung

Abbildung 3.4: Experiment zur Bestimmung der Genauigkeit der Hover Detection.

- die zuletzt erfassten Schwebekoordinaten
- der Zeitpunkt der letzten Erfassung der Schwebekoordinaten

Ob das Ziel tatsächlich berührt wurde, war dabei irrelevant.

Nach jeweils vierzig Durchgängen wurde das Experiment beendet und eine Schaltfläche eingeblendet, mit der der Nutzer wieder auf den Startbildschirm zurückkehren konnte.

Vorgehensweise Die Teilnehmer wurden in zwei Gruppen aufgeteilt, vier führten das Experiment auf dem Samsung Galaxy S4 durch und vier auf dem Samsung Galaxy S5. Die Teilnehmer wurden mündlich instruiert, das Experiment zwei Mal nacheinander durchzuführen. Beim ersten Durchlauf sollten sie darauf achten, das angezeigte Ziel möglichst genau zu berühren, während sie beim zweiten Durchgang versuchen sollten, das Ziel möglichst schnell zu berühren. Damit sollte erreicht werden, dass sowohl langsame als auch schnelle Bewegungen bei der Berührung des Bildschirms verwendet wurden. Während der Durchführung des Experiments war der Versuchsleiter anwesend und gab das Gerät nach Beendigung an den jeweils nächsten Teilnehmer weiter.

Design Bei dem Experiment handelt es sich um ein gemischtes 2×2-Design mit den Leveln *genau* und *schnell* für die unabhängige Variable *Anweisung* sowie den Leveln *Galaxy S4* und *Galaxy S5* für die unabhängige Variable *Gerät*. Innerhalb der Testpersonen wurde die Auswirkung der *Anweisung* gemessen, zwischen den zwei Gruppen der Einfluss der Variable *Gerät*. Als abhängige Variable wurde die Entfernung zwischen Ziel, tatsächlicher Berührung und jeweils letztem erfassten Ziel gemessen. Dabei bezeichnet d_t die Entfernung zwischen der Position des Ziels und dem zugehörigen Touchevent, d_h die Entfernung zwischen der Position dieses Touchevents und des letzten, direkt zuvor empfangenen Hoverevents. Die gemessene Verzögerung t_t zwischen dem Einblenden eines Ziels und der Berührung des Bildschirms sowie die Verzögerung t_h zwischen dem jeweils letzten erfassten Hoverevent und dem darauf folgenden Touchevent eines Ziels dienten als weitere abhängige Variablen, um überprüfen zu können ob die Testpersonen die gegebenen Anweisungen befolgten.

Statistische Auswertung

Vor der statistischen Auswertung wurden die Testdaten mit einer laufenden Nummer zur Identifikation der jeweiligen Testperson erweitert. Außerdem wurden die vorher nur mündlich gegebenen Anweisung (*genau* oder *schnell*) und das jeweils genutzte Gerät mit in den Datensatz geschrieben.

Pro Gerät wurden jeweils 320 Datensätze erfasst (je vier Teilnehmer mit 2×40 Zielen). Allerdings mussten beim Galaxy S4 fünf Datensätze entfernt werden, weil vor der Berührung des Bildschirms keine Hoverposition erfasst wurde. Beim Galaxy S5 gab es 12 solcher Datensätze, die ebenfalls entfernt wurden. Die restlichen 315 bzw. 308 Datensätze wurden zusammengeführt, so dass insgesamt 623 von 640 möglichen Datensätzen analysiert werden konnten. Da gerade das im Alltag zu erwartende Verhalten des Systems samt eventueller Aussetzer und Fehler untersucht werden sollte, wurden keine weiteren Ausreißer identifiziert bzw. entfernt.

Anschließend wurden die Entfernungen zwischen den angezeigten Zielkoordinaten und der tatsächlichen Berührung (d_t) sowie der Berührungsposition und der letzten erfassten Hoverposition (d_h) anhand der Pixel-Koordinaten bestimmt und in Millimeter umgerechnet. Ebenso wurde die vergangene Zeit zwischen dem Anzeigen der Zielkoordinaten und der Berührung des Bildschirms (t_t) sowie der Verzögerung zwischen dem letzten empfangenen Hoverevent und der Berührung (t_h) jeweils in Millisekunden berechnet. Die sich ergebenden Mittelwerte und Standardabweichungen werden in Tabelle 3.2 aufgelistet, eine grafische Darstellung der festgestellten Distanzen zwischen Ziel und tatsächlicher Berührung findet sich in Diagramm 3.6.

Mittels einer Varianzanalyse wurde zunächst überprüft, ob die gegebene Anweisung eine Auswirkung auf die Geschwindigkeit der Teilnehmer beim Berühren des Ziels

Gerät	Variable	Setting	Durchschnitt
Galaxy S4	t_t	*genau*	1124 ms ± 236 ms
		schnell	623 ms ± 138 ms
	t_h	*genau*	37 ms ± 136 ms
		schnell	34 ms ± 122 ms
	d_t	*genau*	1,30 mm ± 0,95 mm
		schnell	2,11 mm ± 1,28 mm
	d_h	*genau*	4,02 mm ± 7,61 mm
		schnell	16,95 mm ± 13,36 mm
Galaxy S5	t_t	*genau*	809 ms ± 234 ms
		schnell	547 ms ± 116 ms
	t_h	*genau*	72 ms ± 185 ms
		schnell	155 ms ± 221 ms
	d_t	*genau*	1,66 mm ± 0,88 mm
		schnell	2,45 mm ± 6,16 mm
	d_h	*genau*	9,27 mm ± 14,66 mm
		schnell	22,13 mm ± 18,12 mm

Tabelle 3.2: Ergebnisse der Messung der Genauigkeit der Hover Detection

hatte. Sie ergab, dass ein signifikanter Unterschied zwischen den Mittelwerten der Verzögerung von Einblenden des Ziels bis zur Berührung des Bildschirms je nach Anweisung existierte ($F(1,6) = 49,966; p \ll 0,05$). Außerdem wurde überprüft, ob die Anweisung einen Einfluss auf die gemessene Zeitspanne zwischen dem Erfassen des letzten Hoverevents vor Berührung des Bildschirms und der dann folgenden Berührung hatte. Hier konnte kein Zusammenhang identifiziert werden ($F(1,6) = 1,351; p > 0,05$).

Ebenfalls mit einer Varianzanalyse wurde untersucht, ob die gegebene Anweisung und die gemessenen Entfernungen zwischen den Positionen des Ziels und der Berührung des Bildschirms voneinander abhängen. Das Ergebnis zeigt, dass es nur einen schwachen Zusammenhang gibt ($F(1,6) = 5,149; p \sim 0,06$). Interessanter ist die Entfernung zwischen der Stelle des letzten Hoverevents und der darauf folgenden Berührung des Bildschirms, hier hatte die erteilte Anweisung eine deutliche Auswirkung ($F(1,6) = 13,899; p < 0,05$). Die Chance, dass die gemessenen Unterschiede zufällig und nicht vom jeweiligen Gerät abhängig sind, beträgt immerhin ca. 25% ($F(1,6) = 1,657; p \sim 0,25$). Wir können daher davon ausgehen, dass das verwendete Gerät keine systematische Auswirkung auf die gemessenen Zeiten hatte.

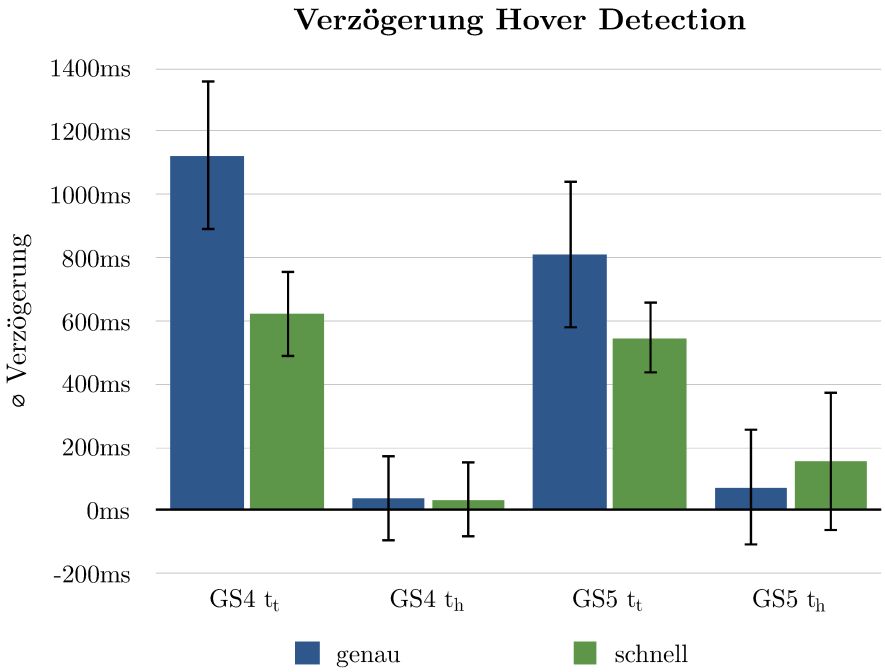


Abbildung 3.5: Latenz in Abhängigkeit vom Setting bei der Messung der Genauigkeit der Hover Detection

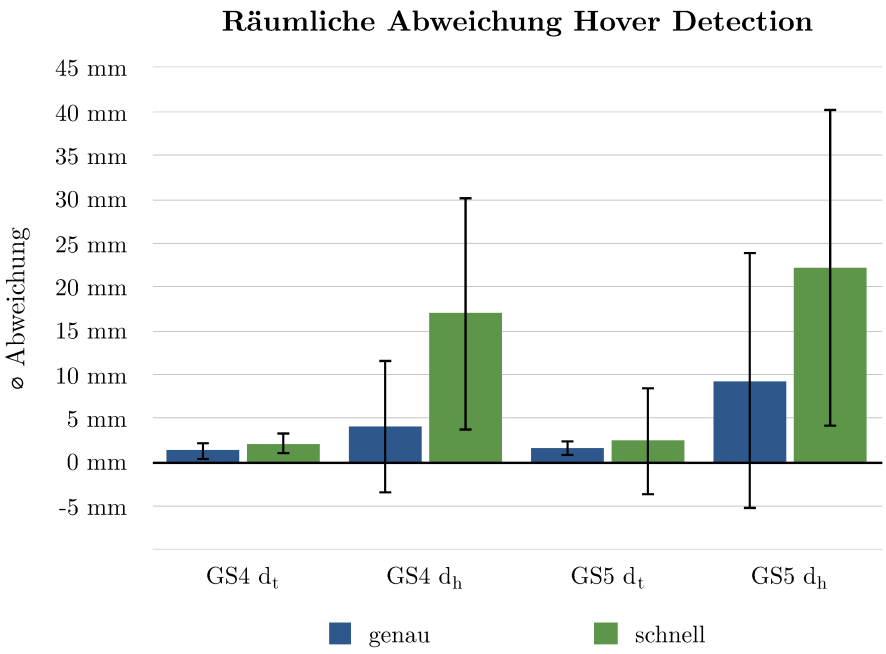
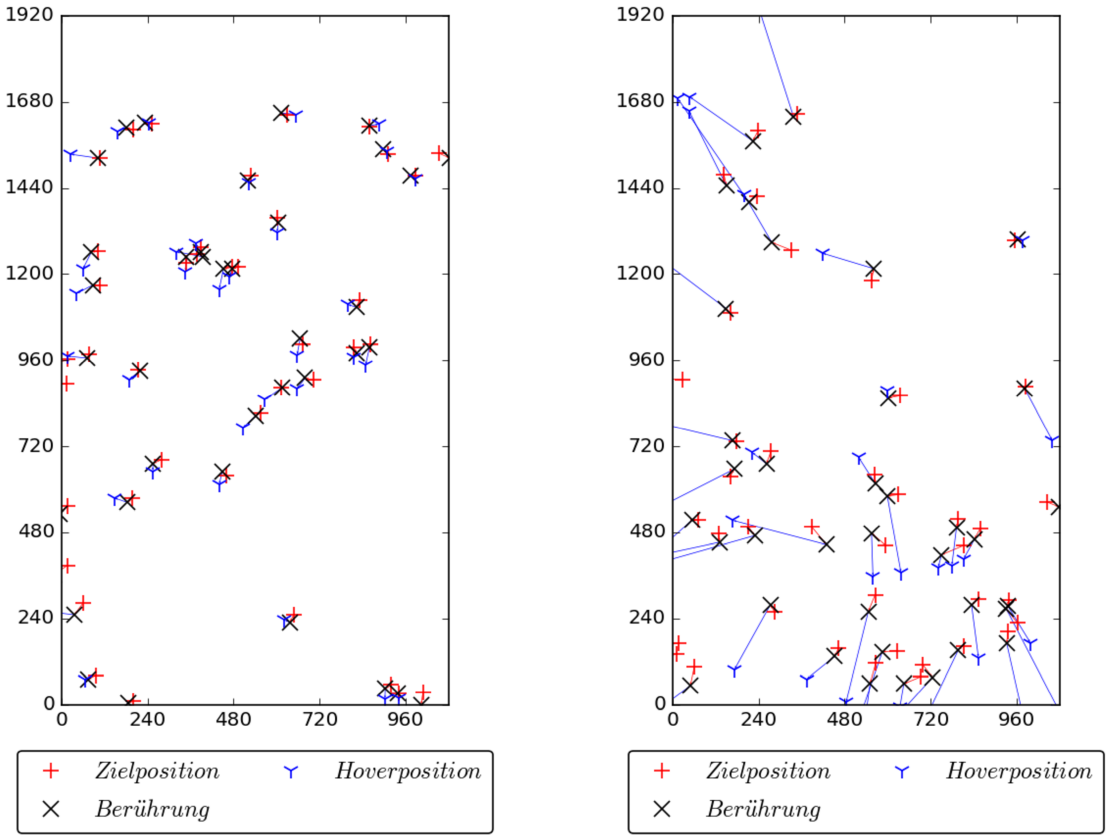


Abbildung 3.6: Räumliche Abweichung zwischen Hover- und Touchevents



(a) Setting *genau*

(b) Setting *schnell*

Abbildung 3.7: Beispiel-Ergebnis zweier Durchläufe von Experiment 3.3.2

Diskussion der Ergebnisse

Das Ziel, mit der Anweisung *genau* oder *schnell* unterschiedliches Verhalten der Teilnehmer hervorzurufen, wurde erreicht. Die durchschnittliche Zeit t_t vom Anzeigen bis zur Berührung eines Ziels ist im Setting *schnell* deutlich geringer als im Setting *genau* (vgl. Diagramm 3.5). Allerdings scheint die gegebene Anweisung keinen großen Einfluss auf die Treffergenauigkeit der Teilnehmer gehabt zu haben, da die Entfernung zwischen angezeigtem Ziel und der darauf erfolgten tatsächlichen Berührung d_t im Setting *genau* zwar kleiner ist als im Setting *schnell*, die Unterschiede aber auch durch Zufall erklärt werden können. Beide Entfernungen sind relativ klein und zeigen, dass die Teilnehmer problemlos dazu in der Lage waren, die angezeigten Ziele mit hoher Genauigkeit zu berühren.

Die durchschnittliche Abweichung zwischen den beim Absenken des Fingers erfassten jeweils letzten Hover- und darauf folgenden Touchpositionen d_h ist größer als die der reinen Touchabweichung und beträgt je nach Gerät durchschnittlich 4 mm (Galaxy S4) bzw. 9 mm (Galaxy S5), wenn die Teilnehmer versuchten das Ziel genau zu treffen. Wenn die Versuchspersonen das Ziel so schnell wie möglich treffen sollten, steigt dieser Wert deutlich an und beträgt dann 17 mm (Galaxy 4) bzw. 22 mm (Galaxy S5). Dieser Anstieg ist so deutlich, dass er nicht mit dem Zufall erklärbar ist. In beiden Settings ist in Diagramm 3.6 gut sichtbar, dass die gemessenen Werte einer hohen Varianz unterworfen sind. Obwohl die beiden untersuchten Geräte im Mittel leicht unterschiedliche Abweichungen lieferten, so sind die Unterschiede dennoch nicht groß genug, um eines der Geräte als deutlich besser oder schlechter zu klassifizieren.

Abbildung 3.7 zeigt, dass die von der Hover Detection gelieferten Werte teilweise deutlich „hinterher hängen“. Die Abbildung zeigt beispielhaft anhand der Daten einer Versuchsperson auf einem Galaxy S4, wo sich auf dem Bildschirm die angezeigten Ziele befanden, welcher Punkt beim versuchten Berühren dieses Ziels tatsächlich berührt wurde und welche Koordinaten von der Hover Detection als letzte Schwebeposition des Fingers gemeldet wurde, bevor es zur Berührung kam. Im Setting *schnell* ist diese Abweichung deutlich erkennbar größer als im Setting *genau*.

Das Experiment demonstriert, dass die Latenz der Hover Detection in alltäglichen Aufgaben einen deutlichen Einfluss auf die Genauigkeit der gelieferten Positionen haben kann. Solange der Nutzer allerdings keine zu hektischen Bewegungen vornimmt, erscheint die Genauigkeit gut genug für die Verwendung zur Einblendung von ortsabhängigem visuellen Feedback. Zwischen den Geräte-Generationen bestehen laut den Messwerten zwar leichte Unterschiede bei den Eigenschaften der Hover Detection, allerdings sind diese nicht groß genug um die Ergebnisse späterer Studien systematisch abhängig vom verwendeten Gerät zu machen.

3.4 Untersuchung von Hover Detection in Zeigeaufgaben

Wie beeinflußt Hover Detection die Leistung in Zeigeaufgaben? Kann visuelles Feedback dabei helfen, kleine Ziele schneller oder genauer zu treffen? Und wenn ja – gilt dies auch für Bildschirmtastaturen? Im folgenden Kapitel werden Experimente vorgestellt, die zur Beantwortung dieser Fragen durchgeführt wurden.

3.4.1 Einfluss von Hover Detection auf Geschwindigkeit und Genauigkeit in Zeigeaufgaben

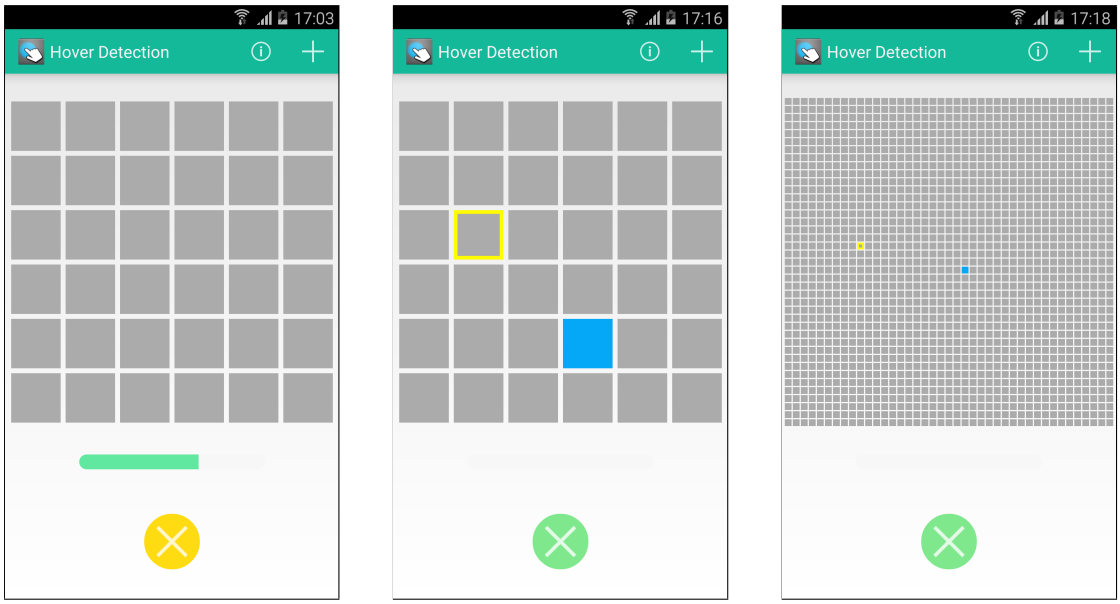
Um zu untersuchen, ob mittels Hover Detection die Durchführung von schwierigen Zeigeaufgaben beschleunigt werden kann, wurde zusammen mit Jan-Hendrik Wolf in seiner Bachelor-Arbeit (Wolf 2014) ein weiteres Experiment konzipiert[2].

Methodik

Als Ausgangssituation wurde angenommen, dass ein sehr kleines Ziel auf dem Bildschirm berührt werden soll. Je kleiner ein Ziel wird, desto schwieriger und damit langsamer wird es nach Fitts' Gesetz (vgl. 2.1.2), dieses Ziel zu berühren. Mit einer für diesen Zweck entwickelten App wurde untersucht, ob die auf Hover-Daten basierende Markierung eines unter dem Finger befindlichen Ziels hilft, ein Ziel einfacher oder schneller berühren zu können als ohne eine solche Markierung. Dies würde erlauben, bei der Verwendung von kleinen Zielen die nach *Fitts' Law* zu erwartenden Performanceeinbußen ganz oder zumindest teilweise ausgleichen zu können. Nachdem in ersten Tests auffiel, dass die angezeigte Hoverposition teilweise von der folgenden Touchposition abwich, wurde zudem ein Mechanismus zur Korrektur der Touchposition implementiert.

Teilnehmer An dem Versuch nahmen 18 Teilnehmer zwischen 20 und 35 Jahren teil, von denen acht weiblich waren. Die Testpersonen waren Mitarbeiter und Studenten an der Universität Bremen und erfahren im Umgang mit Computern und mobilen Geräten wie Tablets oder Smartphones.

[2] Die Programmierung des Prototypen sowie die Durchführung des Experiments und das Erfassen der Messdaten wurden von Jan-Hendrik Wolf ausgeführt. Die Modellierung der Parameter für Fitts' Gesetz sowie die Auswertung der Ergebnisse des SUS wurde von Jan Wolf übernommen, die fachliche Konzeption des Experiments, die statistische Analyse der Überlebenswahrscheinlichkeit, die Interpretation der statistischen Daten sowie die Aufbereitung des Experiments stammen vom Autor dieser Dissertation.



(a) Starten eines Durchgangs (b) Maximale Zielgröße (c) Minimale Zielgröße

Abbildung 3.8: Ziel-Darstellung in Experiment 3.4.1. Blau dargestellt ist das aktuelle Ziel, gelb umrandet das aktuell unter dem Finger befindliche Quadrat. Unten grün die Schaltfläche zum Starten des Durchgangs.

Apparat Als Versuchsapparat diente eine Anwendung, die auf dem Smartphone Galaxy S4 (vgl. 3.2.2) lief. Sie zeigte für das Experiment ein Gitter grauer Quadrate an. Eines der Quadrate wurde von der Anwendung zufällig als Ziel für eine Berührung ausgewählt. Die äußeren Abmessungen des Gitters waren fest, die Größe und damit die Anzahl der Quadrate im Gitter hingegen war variabel, um verschiedene Schwierigkeitsgrade abbilden zu können (siehe Abb. 3.8). Initial betrug die Kantenlänge eines Quadrats 10 mm. Mit jedem Durchgang des Experiments wurde die Kantenlänge der Quadrate verringert, bis sie in der schwierigsten Stufe nur noch 1,4 mm betrug, was nach 26 Durchgängen erreicht war.

Zum Starten eines Durchgangs des Experiment mussten die Teilnehmer eine Schaltfläche für 1,5 s gedrückt halten. Währenddessen wurde ein Fortschrittsbalken eingeblendet, der das Verstreichen der Wartezeit grafisch darstellte. Nach Ablauf dieser Zeit wurde eines der im Gitter dargestellten Quadrate blau eingefärbt und damit als Ziel markiert. Für jedes Ziel wurde der *Index of difficulty* (vgl. Abschnitt 2.1.2) berechnet und aufgezeichnet.

Ab dem Zeitpunkt des Abhebens des Fingers von der Startschaltfläche wurde die Zeit erfasst, die der Nutzer bis zum Berühren des Ziels benötigte. Bei aktivierter Hover Detection wurde zusätzlich das jeweils unter dem Finger befindliche Quadrat mittels eines gelben Rahmens hervorgehoben. Berührte der Nutzer das Ziel, so wurde der nächste Durchlauf des Experiments mit kleineren Zielen gestartet. Falls der Nutzer den Bild-

schirm berührte ohne dabei das Ziel zu treffen, so wurde das Durchgang neu gestartet – der Nutzer musste erneut den Startschalter für 1,5 s berühren, bevor das Ziel eingeblendet wurde. Größe und Position des Ziels blieben gleich, bis es entweder erfolgreich berührt wurde oder der Nutzer fünfmal hintereinander nicht das Ziel berührte. Dies führte zum Abbruch des Experiments, da die (freiwillig teilnehmenden) Testpersonen nicht mit immer schwierigeren, aber für sie nicht mehr lösbaren Aufgaben frustriert werden sollten.

Zusätzlich aktivierbar war ein Korrekturmodus, der etwaige Unterschiede zwischen Hover- und Touchposition ausglich. In diesem Modus wurde die tatsächliche Position der Berührung ignoriert und durch die letzte aufgezeichnete Hoverposition ersetzt, so dass immer genau das Quadrat als berührt galt, das zu diesem Zeitpunkt auch grafisch mit einer gelben Umrandung hervorgehoben wurde. Allerdings bestand aufgrund der Latenz der Hover Detection die Gefahr, dass Hover- und Touchposition stark divergierten und der Nutzer durch die „Korrektur“ verwirrt werden würde, wenn beide Positionen merklich voreinander abwichen. Daher wurde sie nur angewendet, wenn die Distanz zwischen der Hover- und der Touchposition weniger als 7 mm betrug.

Ob Hover Detection und Korrekturmodus aktiv waren, konnte durch Auswahl eines Testmodus bestimmt werden. Bei allen Durchläufen wurden außer dem gerade aktuellen Modus auch eine Nummer zur Identifizierung der Versuchsperson und die Position, Größe und Zeit bis zum Berühren des Ziels sowie die Anzahl der benötigten Versuche pro Schwierigkeitsgrad für die spätere Auswertung aufgezeichnet.

Vorgehensweise Die Untersuchung wurde in Räumlichkeiten der Universität Bremen durchgeführt. Nach einer kurzen Einführung in die untersuchten Themen wurden die Teilnehmer über den weiteren Verlauf des Experiments informiert. Anschließend füllten sie einen Fragebogen zur Erfassung der personenbezogenen Daten aus.

Anhand einer kleinen Beispielanwendung konnten die Teilnehmer Interaktion mit Hover Detection ausprobieren. Sobald sie bereit waren, wurde der erste Test gestartet. Die Teilnehmer wurden angewiesen, nur einen einzelnen Finger zum Berühren der Ziele zu verwenden.

Jeder Teilnehmer durchlief die Aufgabe drei Mal: Ohne graphisches Feedback der Hoverposition, mit grafischem Feedback ohne Korrektur und mit grafischem Feedback und mit angewandter Korrektur. Die Reihenfolge dieser Modi konnte in der Anwendung verändert werden, um eine Ausbalancierung von Lerneffekten zu erreichen. Welche Feedback-Modi es gab und in welcher Reihenfolge sie aktiv waren, wurde den Nutzern vor Durchführung des Experiments nicht mitgeteilt, um unverfälschte Meinungen abfragen zu können.

Nach jedem Durchlauf bekamen die Testpersonen zwei Fragebögen ausgehändigt und füllten sie aus. Um die Gebrauchstauglichkeit des Systems zu erfassen, wurde der *Sys-*

tem Usability Scale (Bangor, Kortum und Miller 2008; Brooke 1996) in einer deutschen Fassung nach Lohmann (2013) eingesetzt. Mit einem weiteren Fragebogen wurden die Teilnehmer sowohl nach positiven als auch negativen Aspekten ihrer Erfahrung gefragt.

Design In dem Experiment wurden als unabhängige Variablen die Schwierigkeit in Abhängigkeit von der Größe eines Ziels mit 26 linearen Abstufungen von 10 mm bis herab zu 1,4 mm Größe sowie das Hover-Feedback mit den drei Abstufungen *nicht aktiv*, *grafisches Feedback ohne Korrektur* und *grafisches Feedback mit Korrektur* manipuliert. Da die Versuchspersonen alle 26 Schwierigkeitsgrade jeweils mit und ohne Feedback durchliefen, handelt es sich um ein 26×3-*within-subjects*-Design. Zum Ausgleichen von Lerneffekten wurden die Reihenfolge der durchlaufenen Modi der Variable *Hover-Feedback* über die Teilnehmer ausbalanciert.

Als abhängige Variablen wurden die Zeit bis zum Berühren eines Ziels und die Anzahl der benötigten Versuche erfasst.

Statistische Auswertung der Experimentaldaten

Insgesamt wurden 1404 Datensätze ausgewertet (18 Teilnehmer $\times$ 3 Modi $\times$ 26 Schwierigkeitsgrade). Allerdings entsprachen nicht alle Datensätze auch einem tatsächlichem Ergebnis einer Testperson, da beim Nicht-Erreichen eines Ziels alle folgenden, schwereren Durchgänge dieser Testperson in diesem Modus als *nicht erreicht* markiert wurden.

Ausreißer-Entfernung Von den tatsächlich vorhandenen Werten wurden solche als Ausreißer entfernt, die mehr als das Doppelte des Interquartilsabstands kleiner als Q1 oder größer als Q3 waren. Nach der Entfernung der Ausreißer blieben 1241 Werte übrig, die für die weitere Analyse herangezogen wurden.

Modellierung der Parameter für Fitts' Gesetz Zur Modellierung der Parameter a und b in Fitts' Gesetz wurde für jeden der 1241 Datensätze zunächst der Schwierigkeitsindex I_d berechnet (vgl. Kapitel 2.1.2):

$$I_d = \log_2 \frac{2A}{W_s} \qquad \text{Index of Difficulty}$$

Dann wurde für jede der einzelnen Testpersonen für jeden durchlaufenen Modus mittels einer linearen Regression die Parameter a und b für Fitts' Formel berechnet:

Feedback	Mittelwert
keines	164 ms ± 87 ms
grafisch ohne Korrektur	440 ms ± 493 ms
grafisch mit Korrektur	493 ms ± 300 ms

Tabelle 3.3: Werte des b-Parameters bei Modellierung von Fitts' Gesetz anhand der Ergebnisse des Experiments 3.4.1

$$t = a + b \cdot I_d = a + b \cdot \log_2 \frac{2A}{W_s} \qquad \text{Fitts' Gesetz}$$

Die sich ergebenden 54 Modelle wurden anhand des Hover-Feedback-Modus gruppiert. Da mit dem Experiment nicht die individuellen Reaktionsgeschwindigkeiten der Testpersonen untersucht werden sollte, wurde der Parameter a in der weiteren Analyse nicht berücksichtigt. Mittelwerte und Standardabweichungen des Parameters b sind in Tabelle 3.3 aufgeführt. Anschließend wurden b mit einer Varianzanalyse auf statistische Zusammenhänge zwischen der Geschwindigkeit der Zielberührung und des aktivierten Feedback-Modus untersucht. Sie ergab, dass der b-Parameter signifikant vom Hover-Feedback-Modus abhing ($F(2, 34) = 16,92; p < 0,05$). Paarweise t-Tests der Ergebnisse der einzelnen Modi ergaben, dass die Unterschiede zwischen dem Modus *kein grafisches Feedback* und jeweils dem Modus *grafisches Feedback ohne Korrektur* und *grafisches Feedback mit Korrektur* existierten.

Überlebensabschätzung Im Modus *Grafisches Feedback mit Korrektur* gelang es immerhin 13 Testpersonen, alle Schwierigkeitsgrade erfolgreich zu bewältigen, gefolgt vom Modus *grafisches Feedback ohne Korrektur* mit elf erfolgreichen Nutzern. Ohne grafisches Hover-Feedback schafften es nur fünf Testpersonen, die Aufgabe komplett zu lösen. Abbildung 3.9 und Tabelle 3.4 zeigen, wie viele Testpersonen jeweils einen Schwierigkeitsgrad erfolgreich absolvierten.

Mit einer Überlebensabschätzung nach Kaplan und Meier (1958) wurde anhand der jeweils als letztes erfolgreich absolvierten Schwierigkeitsgrade untersucht, ob es einen Zusammenhang zwischen dem Erfolg beim Lösen der Aufgabe und dem jeweils aktivierten Feedback-Modus gab. Das Ergebnis zeigt, dass es für die erfolgreiche Absolvierung aller Schwierigkeitsgrade einen signifikanten Unterschied machte, welcher Feedback-Modus aktiv war ($\chi^2 = 8,3$ bei zwei Freiheitsgraden, $p < 0,05$). Die Verwendung eines Modus mit grafischem Feedback führte mit höherer Wahrscheinlichkeit zu einem erfolgreichem Absolvieren aller Schwierigkeitsstufen als der Modus ohne ein solches Feedback.

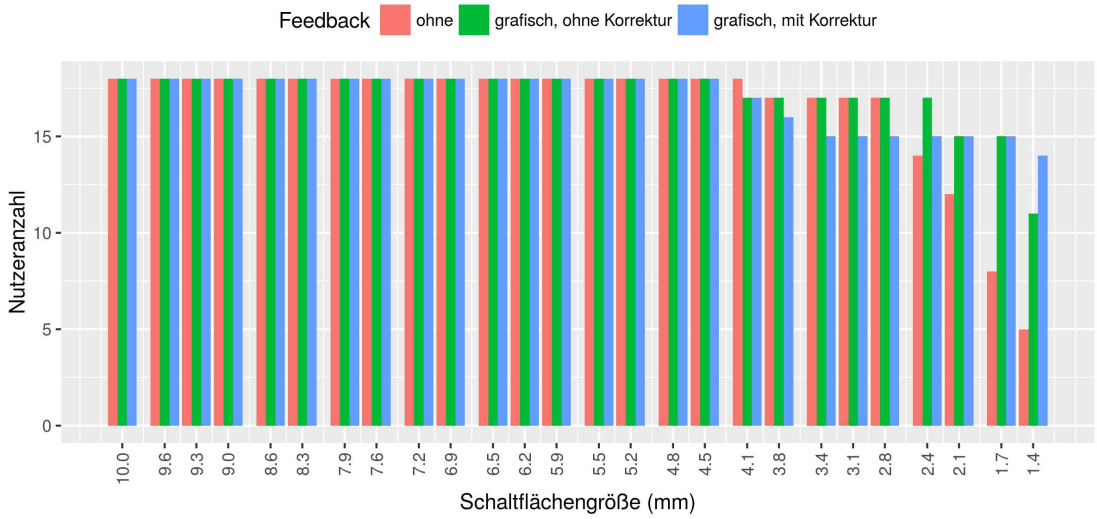


Abbildung 3.9: Anzahl der Versuchspersonen, die die Aufgabe aus 3.4.1 je Feedback-Modus und Zielgröße lösen konnten.

Größe\Feedback	ohne	grafisch, ohne Korrektur	grafisch, mit Korrektur
10 mm	18	18	18
...	...	...	...
4,5 mm	18	18	18
4,1 mm	18	17	17
3,8 mm	17	17	16
3,4 mm	17	17	15
3,1 mm	17	17	15
2,8 mm	17	17	15
2,4 mm	14	17	15
2,1 mm	12	15	15
1,7 mm	8	15	15
1,4 mm	5	11	14

Tabelle 3.4: Anzahl der Versuchspersonen, die die Aufgabe aus 3.4.1 je Feedback-Modus und Zielgröße lösen konnten.

Statistische Auswertung des *System Usability Scale*

Mit dem *System Usability Scale* (SUS) wurde untersucht, wie die Nutzbarkeit der Anwendung bei der Lösung der gestellten Aufgabe eingeschätzt wurde.

Insgesamt wurden der Prototyp unabhängig vom gerade aktiven Feedback-Modus relativ gut bewertet, alle Scores liegen zwischen 70 und 80 Punkten. Ohne grafisches Feedback wurde im Mittel ein Score von $73,6$ Punkten vergeben, mit grafischem Feedback ohne Korrektur waren es $75,8$ und mit Feedback und Korrektur $73,3$ Punkte. Eine Varianzanalyse der Werte ergab, dass die gemessenen Unterschiede mit hoher Wahrscheinlichkeit zufällig verursacht waren ($F(2,34) = 0,246; p \sim 0,78$). Der aktive Feedback-Modus hatte damit keinen Einfluss auf die Nutzbarkeit des Prototypen.

Generell gilt die Nutzbarkeit ab einem SUS-Score von 68 Punkten als überdurchschnittlich. Nach Bangor, Kortum und Miller (2008) entsprechen die hier erzielten Bewertungen einer verbalen Beschreibung von *gut* bis *exzellent*.

Diskussion der Ergebnisse

Bei der Auswertung der Ergebnisse fällt zunächst auf, dass alle Teilnehmer des Experiments es bis zu einer Zielgröße von 4,5 mm schafften, die angezeigten Ziele zuverlässig zu berühren. Dies kann als Bestätigung der Interface-Richtlinien der großen Mobilbetriebssysteme gesehen werden, deren empfohlene Mindestgrößen für interaktive Elemente alle über diesem Wert liegen.

Die meisten der Teilnehmer konnten auch Ziele, die zwischen 4,5 mm und 2,4 mm Kantenlänge hatten, noch gut treffen. Bei weiterer Verringerung der Zielgröße schafften dies immer weniger Teilnehmer. Bis zu einer Größe von 2,4 mm machte es dabei keinen großen Unterschied, ob einer der grafischen Feedback-Modi aktiv war.

Erst bei einer Zielgröße von 2,1 mm und kleiner zeigte sich, dass die Einblendung der aktuellen Hoverposition den Teilnehmern dabei half, die Ziele zu berühren. Bei der kleinsten Zielgröße von 1,4 mm schafften es nur fünf der achtzehn Teilnehmer das Ziel zu berühren, wenn kein grafisches Feedback eingeblendet wurde. Bei Anzeige der aktuell erkannten Hoverposition war es sechs weiteren Teilnehmern möglich das Ziel zu berühren. Wurde die Berührungsposition zusätzlich auf die zuletzt erfasste Hoverposition verschoben und so eine räumliche Kongruenz zwischen den beiden Positionen geschaffen, konnten immerhin 14 der 18 Testpersonen selbst die kleinste Zielgröße noch erfolgreich berühren.

Die Ergebnisse des Experiments zeigen, dass eine visuelle Anzeige der aktuellen Fingerposition Nutzern helfen kann, selbst sehr kleine Ziele auf einem Smartphone noch berühren zu können. Dabei muss allerdings je nach Größe des intendierten Ziels berücksichtigt werden, dass Hoverposition und direkt folgende Berührungsposition leicht

voneinander abweichen können. Eine Korrektur ist möglich und wird von den Nutzern nicht negativ bemerkt, wie die Ergebnisse des SUS zeigen.

3.5 Hover Detection für kontextsensitives Feedback in Bildschirmtastaturen

Wie Experiment 3.4.1 zeigte, kann Hover Detection Nutzern dabei helfen, kleine Ziele auf einem Smartphone besser zu treffen. Allerdings ist die Problematik zu kleiner Ziele nicht neu. Aus diesem Grund veröffentlichen die Hersteller von Mobilbetriebssystemen Richtlinien, wie grafische Oberflächen auf ihren Systemen aussehen sollen. In diesen ist in der Regel auch festgehalten, wie groß ein interaktives Element mindestens sein sollte, damit es zuverlässig berührt werden kann.

Es gibt jedoch auf fast jedem Smartphone ein Beispiel, bei dem diese Regeln nicht eingehalten werden können: Die Bildschirmtastatur. Sie ist auf Mobilgeräten heute das am meisten verwendete Mittel zur Texteingabe. Damit alle von einer klassischen Tastatur benötigten Tasten gleichzeitig dargestellt werden können, müssen diese relativ klein sein. Das macht sie sowohl schwerer zu treffen als auch anfälliger für Verdeckung mit dem Finger, der für die Berührung des Bildschirms verwendet wird. Für Nutzer mit alters- oder gesundheitsbedingten Einschränkungen können diese Probleme noch deutlich stärker zu Tage treten.

Kann Hover Detection bei diesen Problemen helfen? Zur Untersuchung dieses Themenkomplexes wurde ein Tastatur-Prototyp entwickelt, der auf einen schwebenden Finger reagierte. Für die Evaluation wurden sowohl ältere Teilnehmer als auch Testpersonen mit visuellen Einschränkungen ausgewählt.

Da die zu implementierende Tastatur Hover Detection und die Vergrößerung der Tastaturdarstellung unterstützen sollte, lag der Projektname nahe: *HoverZoom-Keyboard*.

3.5.1 Zielgruppe

Eine der Zielgruppen, die von größeren Tasten besonders profitieren könnte, sind ältere Nutzer aufgrund häufig beeinträchtigter Sehfähigkeit (vgl. Abschnitt 2.1.4). Neben der Einschränkungen des Visus treten in dieser Altersklasse auch häufig eine verminderte Feinmotorik auf, auch hier könnte aus größeren Tasten ein Vorteil gezogen werden. Beginnenden kognitiven Einschränkungen könnte durch Rückgriff auf existierende Vorkenntnisse sowie eine intuitive Nutzbarkeit begegnet werden.

Nach ersten Gesprächen mit potentiellen Probanden der Altersklasse 65+ wurde schnell klar, dass das für diesen Prototypen geplante visuelle Feedback ohne eine interaktive Demonstration nicht vollkommen verstanden werden würde. Da es sich bei dem Prototypen um eine Tastatur handeln sollte, war zudem eines der zu erreichenden Ziele

die Vergleichbarkeit mit existierenden Tastaturlösungen. Damit dieser Vergleich nicht durch die oben angesprochenen Erwartungshaltungen der Nutzer verfälscht werden würde, fiel die Entscheidung, einen *high-fidelity*-Prototypen zu erstellen. Dieser sollte alle bei einer Bildschirmtastatur gängigen Funktionen unterstützen.

3.5.2 Anforderungen

Das Ziel bei der Entwicklung des Prototypen war eine Vergleichbarkeit mit existierenden Bildschirmtastaturen, daher wurde der zu erreichende Funktionsumfang entsprechend festgelegt. Damit sollte sicher gestellt werden, dass bei der Auswahl von Texten für die Tests keine Einschränkungen hinsichtlich der einzugebenden Zeichen zu berücksichtigen wären. Zu diesem Zweck wurden verschiedene für Smartphones verfügbare Bildschirmtastaturen untersucht und gemeinsame Merkmale identifiziert.

Die folgenden Merkmale wurden als „gemeinsamer Nenner“ festgelegt, um eine sinnvolle, vergleichende Evaluation zu ermöglichen:

1. Interaktive Nutzung möglich
2. Gewohnte Darstellungsgröße der Tastatur
3. Tastenanordnung in bekanntem Layout: Eine Tastatur mit klassischem QWERTZ-Layout würde zulassen, auf vorhandenen Kenntnissen mit Schreibmaschinen- oder PC-Tastaturen aufzubauen.
4. Eingabe von Buchstaben
5. Umschaltung zwischen Groß- und Kleinschreibung
6. Eingabe von Umlauten
7. Eingabe von Zahlen
8. Eingabe von Sonderzeichen

Zusätzlich sollte der Prototyp auch die Einblendung visuellen Feedbacks auf die aktuelle Hoverposition eines Fingers erlauben:

9. Hover Detection mit grafischem Feedback möglich

Bewusst nicht als Anforderung aufgenommen wurden Verfahren zur Rechtschreibkorrektur oder zur Vorhersage folgender Worte. Diese Funktionen dienen primär dem Zweck, eine fehlerhaft getätigte Eingabe zu vermeiden oder zu korrigieren. Da mit diesem Prototypen aber gerade untersucht werden sollte, ob die Eingabe als solche verbessert werden könnte, wurden nachgeschaltete Mechanismen außen vor gelassen.

3.5.3 Funktionsumfang

Zur Erfüllung der gestellten Anforderungen wurde eine Anwendung für Android mit den folgenden initialen Eigenschaften entwickelt. Spätere Erweiterungen werden gesondert beschrieben.

Interaktive Nutzung und Evaluation Um die Nutzung der Tastatur als interaktiven Prototypen zu ermöglichen, musste sie auf einem der Geräte lauffähig sein, die mit Hover Detection ausgeliefert wurden. Wie auch bei den Prototypen der bereits beschriebenen Experimente schränkte dies die Auswahl auf das Samsung Galaxy S4, später auch das Galaxy S5 ein (3.2), die beide mit dem Betriebssystem Android laufen.

Android erlaubt Programmierern und Nutzern sehr weitreichende Eingriffe in das Betriebssystem, wie zum Beispiel den Austausch der von Android zur Verfügung gestellten Bildschirmtastatur. Leider stellte sich bei der Implementierung des Tastatur-Prototypen heraus, dass die von *AirView* erzeugten Hoverevents nicht an Systemdienste wie Tastaturen zugestellt werden. Einer der Grundsätze bei der Entwicklung von Android durch Google ist, dass die Systemprogrammierer nur solche Schnittstellen verwenden, die Entwicklern normaler Anwendungen auch zur Verfügung stehen (Meier 2008, S. 3). Allerdings wurden die Quelltexte der von Samsung zusätzlich eingefügten Android-Features wie *AirView* nicht offen gelegt, daher konnte dieses Problem auch nicht behoben werden. Der Prototyp ließ sich daher bei Verwendung der Hover Detection nur als Bestandteil einer eigenständigen *App* realisieren.

Um die Illusion einer tatsächlich funktionierenden Tastatur zu bieten, wurde ein Textfeld in die Anwendung integriert, das bei Betätigung einer Taste die entsprechende Aktion ausführte. Die Interaktion mit diesem Textfeld ist jedoch „fest verdrahtet“, eine Nutzung der Tastatur zur Texteingabe ist nur in dieser Anwendung und für dieses Textfeld möglich.

Zur Unterstützung bei der Evaluation wurde außerdem noch ein optionales zweites Textfeld vorgesehen, in das Vorlagentexte für Kopier-Aufgaben geladen werden können. Mittels einer zuschaltbaren Protokollfunktion kann jeder Tastendruck zusammen mit einem Zeitstempel in einer Datei für spätere Auswertung gesichert werden. Zusätzlich wurde ein Programm für den PC geschrieben, das diese Logdatei nach verschiedenen Kriterien auswerten konnte.

In einigen der im folgenden beschriebenen Versuche wurde der *System Usability Survey* verwendet. Er wurde für die Texteingabe-Experimente in Anlehnung an Bangor (Bangor, Kortum und Miller 2008) (vgl. auch 2.1.2) modifiziert, so dass statt dem Wort *System* das Wort *Tastatur* verwendet wurde.

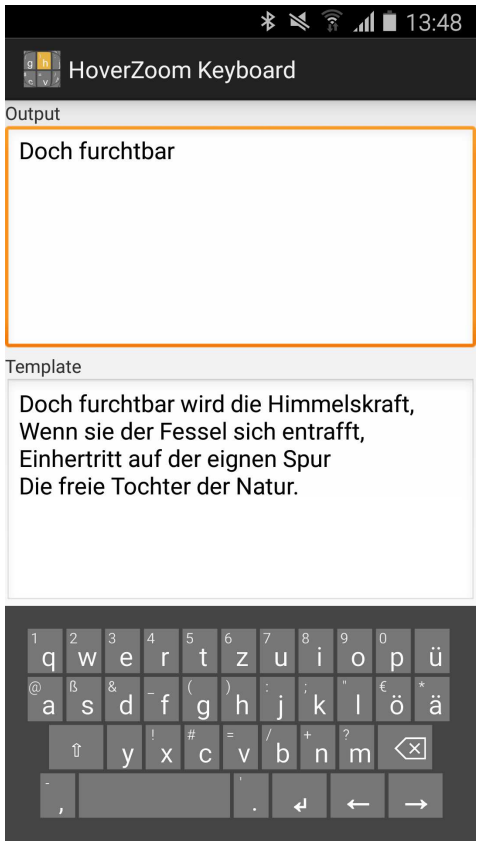


Abbildung 3.10: Bildschirmfotos des *HoverZoom-Keyboard*. Oben das Textfeld, in welchem die getätigten Eingaben erscheinen. In der Mitte das optionale Textfeld für Vorlagentexte. Unten die Tastaturdarstellung mit deutschem Layout.

Darstellung Die Tastatur nimmt im Hochformat ein Drittel der Bildschirmhöhe ein (Abb. 3.10). Wenn das Gerät im Querformat gehalten wird, bedeckt sie die Hälfte des Bildschirms (Abb. 3.11). Allerdings lässt diese Darstellung nicht genug Raum für das Vorlagentextfeld, so dass die Experimente generell im Hochformat durchgeführt wurden.

Die Auswahl und prinzipielle Anordnung der darzustellenden Tasten wird über eine XML-Datei definiert, in der die Tastatur mit Hilfe von Zeilen und der jeweils darin enthaltenen Tasten beschrieben wird. Buchstaben werden standardmäßig als Kleinbuchstaben dargestellt und ausgegeben. Wenn über die Betätigung der Umschalttaste auf die einmalige oder (per langer Berührung) dauerhafte Ausgabe von Großbuchstaben umgestellt wird, sieht der Benutzer dies durch die Verwendung von Großbuchstaben auf der Tastenbeschriftung. Zusätzlich kann einer Taste neben ihrer normalen Funktion auch eine alternative Belegung zugewiesen werden, die über eine lange Berührung der Taste ausgelöst wird. Diese alternative Funktion wird auch auf der Taste in der linken oberen Ecke leicht verkleinert dargestellt.

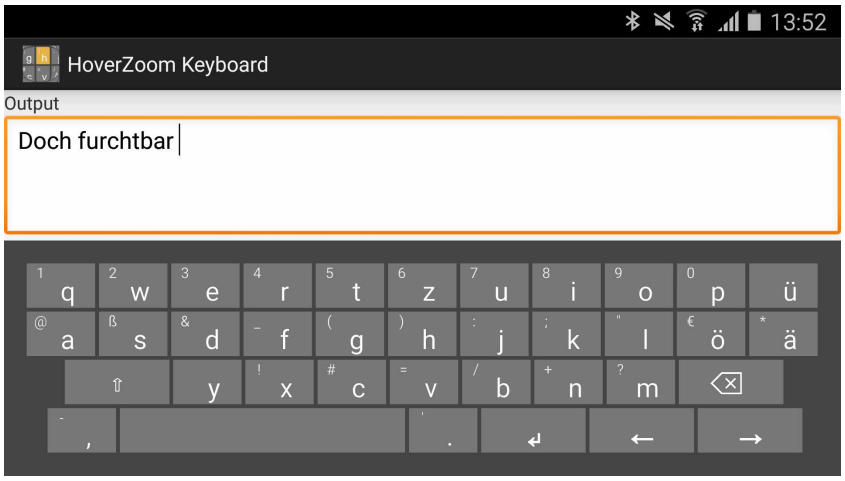


Abbildung 3.11: Die Tastatur im Querformat. Oben die Eingaben, unten die Tastatur. Mangels Platz hier keine Enblendung des Vorlagen-Textfelds.

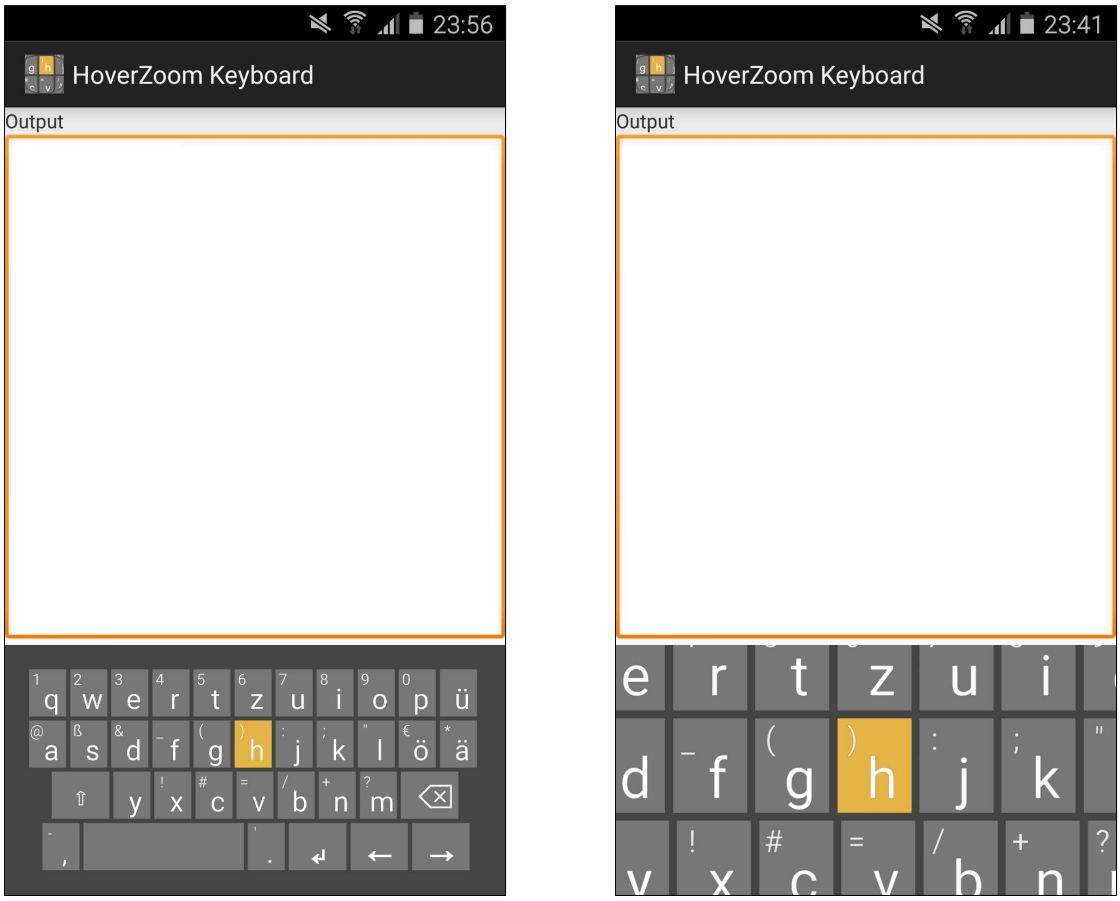
Für einige Spezialtasten wie die Rück- oder die Umschalttaste wurden die zu verwendenden Symbole anhand der Funktion festgelegt, diese sind nicht konfigurierbar.

Die Größe der Tasten auf der verfügbaren Fläche wird automatisch berechnet. Dabei wird anhand der Zeile mit den meisten Tasten bestimmt, wie breit eine Taste unter Berücksichtigung der per XML-Attribut definierbaren Abstände zwischen den Tasten sowie Breite ihrer Umrandung maximal sein darf. Ein zusätzliches Attribut erlaubt es, Tasten ein Vielfaches der so berechneten Standardbreite zuzuweisen. Verwendet wird dies zum Beispiel bei der Leertaste, die typischerweise deutlich breiter ist als eine normale Buchstabentaste. Für die Höhe der Tasten wird die Anzahl der Zeilen herangezogen.

Alle beim Darstellen der Tastatur verwendeten Farben lassen sich als XML-Attribut des Tastaturlayouts festlegen. Die bei den Experimenten verwendeten Farben orientieren sich an dem Farbschema der weit verbreiteten Android-Tastatur *Swype*.

Hover Detection mit grafischem Feedback Als einfachste Variante des grafischen Feedbacks wird die unter dem schwebenden Finger liegende Taste gelb eingefärbt, so dass der Nutzer erkennen kann, welche Taste er beim Absenken des Fingers betätigen würde (Abb. 3.12a).

Ein im Hoverbereich erkannter Finger kann außerdem je nach Konfiguration der Tastatur eine lineare Vergrößerung der Tastaturdarstellung auslösen. Diese verdoppelt den Darstellungsmaßstab ausgehend von der Fingerposition (Abb. 3.12b). Als Folge können jedoch die weiter außen liegenden Bereiche nicht mehr dargestellt werden, da sie aus dem sichtbaren Bereich heraus geschoben werden.



(a) Hervorhebung der Taste *h*.

(b) Lineare Vergrößerung der Tastatur.

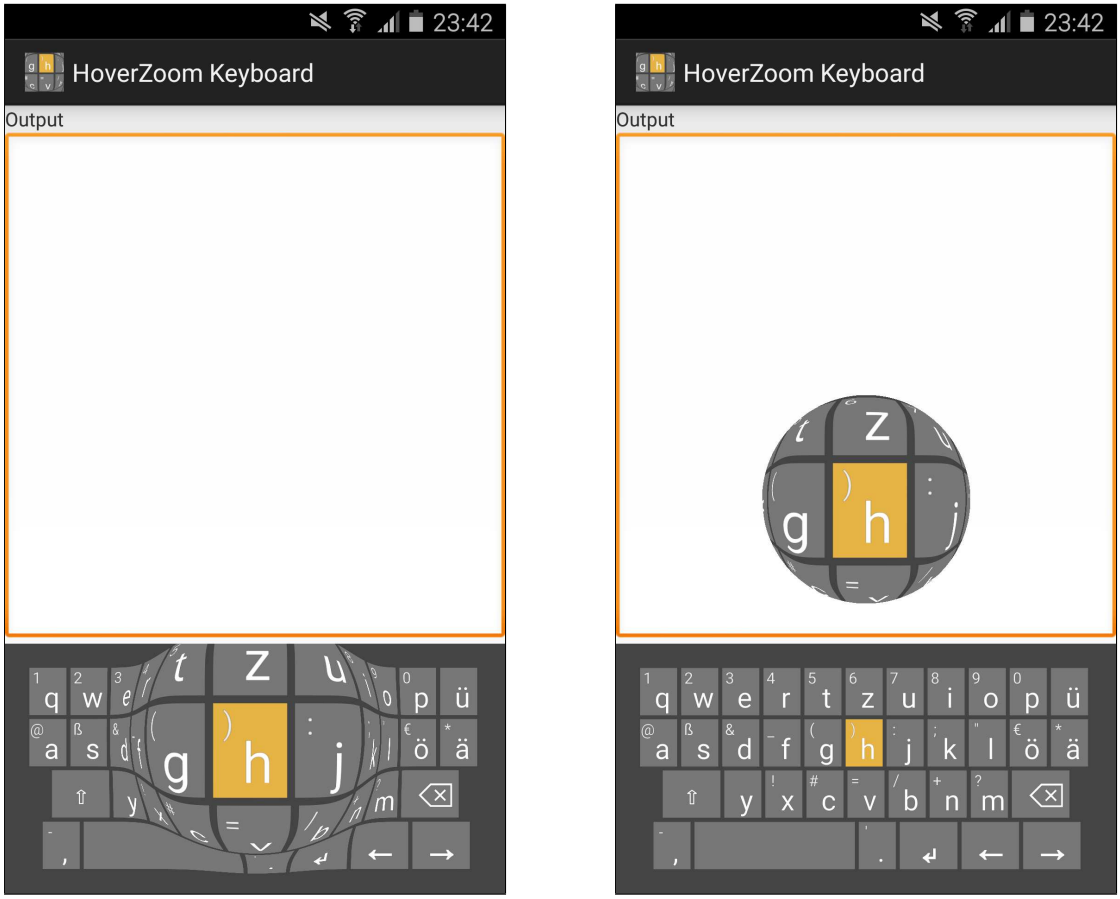
Abbildung 3.12: Grundlegende Hoverfähigkeiten zu Beginn der Tastaturentwicklung.

Um dieses Problem zu lösen wurden zusätzlich in Anlehnung an die Arbeiten von Furnas (1986), Raynal und Truillet (2007) und Vogel und Baudisch (2007) die Darstellung einer Fisheye-Vergrößerung sowohl direkt unter dem Finger (Abb. 3.13a) als auch nach oben versetzt (Abb. 3.13b) implementiert. Die nach oben versetzte Darstellung kann ebenfalls ohne oder mit linearer Vergrößerung verwendet werden.

3.5.4 Technische Umsetzung

Eine grobe Beschreibung der Implementierung befindet sich im Anhang, vor allem in Bezug auf die Verwendung der (A.1) sowie die Umsetzung der Vergrößerungseffekte mit OpenGL ES (A.2).

Eine erste Beschreibung der Funktionen dieser Tastatur wurde als Extended Abstract und Poster bei der Konferenz ACM CHI 2014 eingereicht und akzeptiert (Pollmann, Wenig und Malaka



(a) Fisheye-Vergrößerung unter dem schwebenden Finger.

(b) Nach oben verschobene Fisheye-Darstellung.

Abbildung 3.13: Initiale Fisheye-Vergrößerungen der Tastatur.

2014). Das beim Präsentieren des Posters erhaltene Feedback wurde bei der Weiterentwicklung des Prototypen berücksichtigt.

3.5.5 Untersuchung mit älteren Menschen

In der vom Autor betreuten Diplomarbeit von Möhle (2015) wurde die Verwendung des *HoverZoom-Keyboard* durch ältere Menschen untersucht[3]. Zunächst sollte dabei ein explorativer Ansatz verfolgt werden, um die Vielzahl der möglichen Kombinationen der Vergrößerungsfeatures auf die am besten funktionierenden einzuengen. Im Anschluss sollten diese empirisch evaluiert werden. Basierend auf diesem Plan ergaben sich aufeinander aufbauende Phasen. Die in jeder Phase zu untersuchenden Fragen sowie die gewonnenen Erkenntnisse werden hier kurz beschrieben.

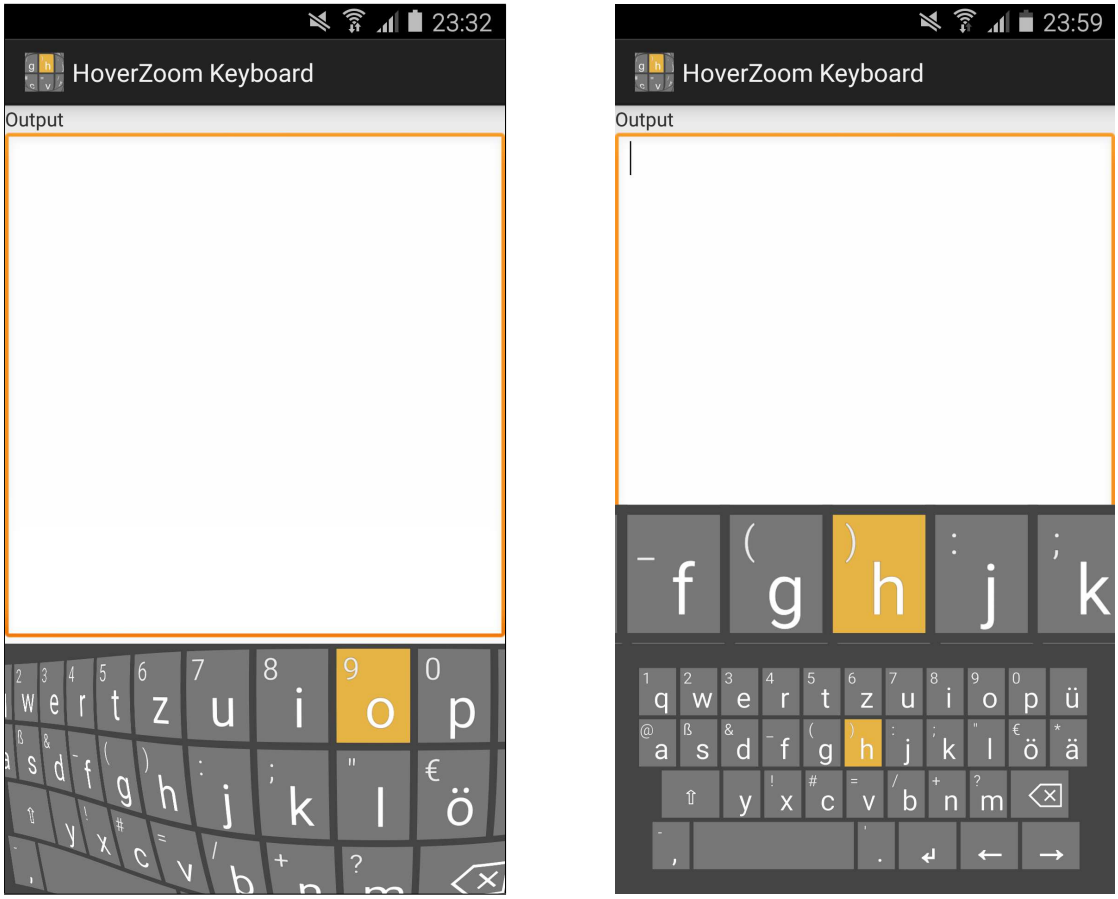
Neue Funktionen

Zu Beginn der Untersuchung wurde zunächst drei zusätzliche Funktionen implementiert:

Große Fisheye-Darstellung Eine prototypische bereits vorhandene Darstellung einer Fisheye-Vergrößerung mit großem Radius (Abb. 3.14a) wurde so überarbeitet, dass sie einen weicheren Übergang zwischen normaler und vergrößerter Darstellung bot.

Vergrößerungsbalken über der Tastatur Die bereits bestehende Funktion der nach oben versetzten Fisheye-Vergrößerung (vgl. Abb. 3.13b) hat den Nachteil, dass sie keine optische Einheit mit der eigentlichen Tastatur bildet. Um dies auszugleichen, wurde ein direkt über der Tastatur befindlicher Balken integriert, der alternativ zu der versetzten Fisheye-Vergrößerung benutzt werden kann. Der Balken liegt direkt über der Tastatur und nutzt die gesamte Bildschirmbreite (Abb. 3.14b). Seine Höhe ist auf 100 dp festgelegt, was bei einem Galaxy S4 1,73 cm entspricht. In dem Balken können wahlweise die bereits erwähnten Darstellungsarten (lineare Vergrößerung, kleine und große Fisheye-Darstellung) verwendet werden.

[3] Die Erweiterungen des *HoverZoom-Keyboard* sowie die Durchführung der Interviews wurden von Christoph Möhle vorgenommen. Die Experimente sowie die Erfassung der Daten wurden gemeinsam geplant und durchgeführt. Die Ergebnisse werden hier kurz und paraphrasiert vorgestellt; für eine detaillierte Wiedergabe der Ergebnisse sei auf die Diplomarbeit an sich verwiesen.



(a) Die überarbeitete große Fisheye-Darstellung.

(b) Über der Tastatur dargestellter Vergrößerungsbalken.

Abbildung 3.14: Erweiterungen der Tastaturdarstellung

Zeilen- und Tastenstabilisierung Besonders im Vergrößerungsbalken fiel auf, dass die Zentrierung der Vergrößerung um die aktuell erfasste Hoverposition des Fingers in einer sehr unruhigen Darstellung resultierte, vor allem aufgrund der in Abschnitt 2.4.1 angesprochenen Bewegungseffekte. Zum Ausgleich wurde eine stabilisierte Darstellung entwickelt. Bei dieser dient die Position des Fingers nur noch zur Feststellung der gerade unter dem Finger befindlichen Tastaturzeile oder -taste. Die Vergrößerung wird auf die Mitte der Zeile bzw. Taste zentriert, so dass kleine Bewegungen des Fingers keine oder nur geringe Auswirkungen auf die Darstellung haben, solange er über derselben Zeile oder Taste bleibt.

Zielgruppeninterview

Um eine erste Meinung aus der Zielgruppe einzuholen, wurde ein Interview mit einem Mitglied der Zielgruppe „ältere Nutzer“ durchgeführt.

Teilnehmer Bei der Gesprächspartnerin handelte es sich um eine 84-Jährige, die dem Interviewer Christoph Möhle persönlich bekannt war. Sie verfügte über keine vorherigen Erfahrungen im Umgang mit einem Computer oder einer Schreibmaschine.

Apparat Zum Demonstrieren der Tastatur wurde diese auf einem Samsung Galaxy S4 installiert und vorgeführt.

Vorgehensweise Die Testperson wurde zuhause besucht. In einem freien Interview, basierend auf Vorgesprächen mit anderen Mitgliedern der Zielgruppe sowie Literaturrecherche, wurden ihre Beweggründe für die Nutzung von Mobiltelefonen und ihre Meinung zu Seniorengeräten erfasst. Nach dem Interview wurde sie gebeten, mit dem Prototypen einen Text zu verfassen.

Ergebnisse Als Motivation für die Nutzung eines Mobilgerätes wurden Sicherheitsgedanken angegeben, etwa das Rufen von Hilfe in Notfällen oder die Absicherung gegen den Ausfall des Festnetztelefons. Weitere Funktionen wurden nicht benötigt.

Die Verwendung der Hover Detection stellte kein Problem dar. Bei der Nutzung der Bildschirmtastatur hatte die Testperson aufgrund der Verdeckung mit dem Finger Probleme, die Tasten richtig zu treffen. Sowohl die abgesetzte Fisheye-Darstellung als auch der Vergrößerungsbalken halfen hier, ebenso die Verwendung eines Stylus. Problematisch war das ungewollte Auslösen der alternativen Tastenbelegungen durch längere Berührung der Tasten. Mangels Vorerfahrungen war die QWERTZ-Anordnung der Buchstaben auf der Tastatur ungewohnt.

Neue Funktionen Basierend auf den Erkenntnissen des Interviews wurden ein Vibrationsfeedback bei der Betätigung einer Taste sowie ein alphabetisches Tastaturlayout programmiert.

Studie 1

Mit dieser Untersuchung sollte in der Vielfalt der Kombinationen die für die Zielgruppe „ältere Nutzer“ am besten funktionierende Zusammenstellung gefunden werden, anhand derer in einer anschließenden Studie ein quantitative, vergleichende Untersuchung durchgeführt werden sollte.

Teilnehmer An der Studie nahmen elf Personen zwischen 53 und 74 Jahren (Durchschnittsalter 64 Jahre) teil. Auch sie waren dem Versuchsleiter Christoph Möhle aus seiner Tätigkeit in der freiwilligen Feuerwehr persönlich bekannt, so dass bereits ein Vertrauensverhältnis bestand und die sich zur Teilnahme bereit erklärten. Eine aus dieser Konstellation potentiell entstehende Voreingenommenheit wurde hingenommen, da eine Rekrutierung von Teilnehmern für diese Studie sonst sehr schwierig geworden wäre. Alle Teilnehmer trugen im Alltag und während der Studie eine Sehhilfe. Bei sechs der Probanden bestand Erfahrung mit einem Smartphone, ein PC wurde bei neun Personen regelmäßig zur Texteingabe verwendet.

Apparat Für die Studie wurde das *HoverZoom-Keyboard* auf einem Samsung Galaxy S4 verwendet.

Vorgehen Die Teilnehmer wurden zuhause besucht. Nach Ausfüllen eines Fragebogens zu persönlichen Daten und Erfahrungen mit Smartphones und PCs wurden die Funktionen des Prototypen vorgestellt. Im Anschluss erhielten die Teilnehmer Zeit, die verschiedenen Funktionen der Tastatur in beliebiger Zusammenstellung selbst auszuprobieren. Die bevorzugten Kombinationen wurden danach in einem Fragebogen erfasst und mit einem *System Usability Scale* sowie einigen offenen Fragen bewertet.

Konfigurierbar waren:

- Tastaturlayout: QWERTZ oder alphabetisch
- Position des grafischen Feedbacks: Auf der Tastatur; nach oben verschobene Darstellung; Vergrößerungsbalken
- Art der Vergrößerung: Linear; kleines Fisheye; großes Fisheye.
- Hervorhebung der Taste unter dem Finger
- Dauer eines optionalen Vibrationsfeedbacks

Ergebnisse Mit Ausnahme eines Teilnehmers verwendeten alle Probanden die Tastatur und stellten sich eine bevorzugte Konfiguration zusammen. Die Bewertung dieser individuellen Kombination mit dem *SUS* ergab, dass es jedem Teilnehmer gelang, eine gut nutzbare Kombination zu erstellen. Der Wertebereich lag auf ganze Zahlen gerundet zwischen 73 und 100 bei einem Mittelwert von 89 ± 7.

Vier der Probanden verwendeten in ihrer bevorzugten Kombination gar keine abgesetzten Darstellungseffekte. Auf der Tastatur selbst wurden von vier Personen Vergrößerungseffekte aktiviert. Drei nutzten das kleine Fisheye, eine Person die lineare Vergrößerung. Fünf Teilnehmer wählten die nach oben versetzte Darstellung, davon zwei ohne Vergrößerung und drei mit dem kleinen Fisheye-Effekt. Nur ein Teilnehmer wählte hier die große Fisheye-Darstellung.

Alle Teilnehmer verwendeten die farbige Hervorhebung der Taste unter dem Finger, einer erwähnte sie explizit positiv im Fragebogen nach der Studie.

Die Stabilisierung wurde von sechs Teilnehmer für die jeweils aktive Taste gewählt. Zeilenstabilisierung wurde gar nicht verwendet.

Das QWERTZ-Layout wurde von sieben Testpersonen gewählt, drei bevorzugten das alphabetische Layout. Bei Verwendung des QWERTZ-Anordung konnte die Verzögerung für das Auslösen der alternativen Tastenbelegung angepasst werden (die alphabetische Anordung hatte keine solchen Belegungen). Hier blieben drei Probanden bei den voreingestellten 500 ms, von weiteren drei wurden 700 ms gewählt. Ein Person bevorzugte sogar eine Verzögerung von 1000 ms.

Acht der Probanden verwendeten Vibrationsfeedback mit einer Verzögerung zwischen 20 ms und 50 ms.

Die Verwendung der Hover Detection stellte für acht Teilnehmer kein Problem dar, während zwei den Erkennungsbereich wiederholt verließen. Einer von diesen berührte auch mehrfach ungewollt den Bildschirm und verrutschte beim Absenken des Fingers auf einen Buchstaben.

Tippfehler traten bei zwei Teilnehmern häufig auf, bei fünf gelegentlich und nur bei drei Testpersonen selten. Eine Korrektur der Fehler gelang sieben Probanden ohne Schwierigkeiten, drei hatten Probleme mit der Platzierung der Schreibmarke auf dem Bildschirm.

Diskussion Jeder der Probanden konnte sich die Tastatur so individuell zusammenstellen, dass er sie gut verwenden konnte. Dabei waren die verwendeten Kombinationen relativ unterschiedlich. Dennoch ließen sich drei Erkenntnisse für Assistenzfunktionen aus der Studie ableiten:

1. Die Tastatur sollte immer ganz zu sehen sein, sonst geht älteren Nutzern schnell die Übersicht verloren.

2. Bei linearer Vergrößerung wirken Bewegungen sehr hektisch.
3. Vibrationsfeedback ist wichtig, um zu erkennen ob und wann eine Taste tatsächlich betätigt wurde.
4. Pfeiltasten können helfen, Textstellen gezielt zu erreichen um beispielsweise fehlerhafte Eingaben zu korrigieren.

Außerdem fiel auf, dass Schreibfehler oft nur deswegen gemacht wurden, weil die Testpersonen die Tastatur in ihrem gewohntem Tempo verwenden wollten und durch langsame, nachziehende Tastatureffekte dabei gestört wurden. Dies bedeutet, dass die in Abschnitt 3.3.1 festgestellte Latenz in der Praxis negative Auswirkungen hat.

Neue Funktionen Nachdem in dieser Studie der Wunsch nach Pfeiltasten zur Kontrolle der Schreibmarkenposition geäußert wurde, erfolgte eine Erweiterung des Tastaturlayouts um diese Fähigkeiten. Mit den Pfeiltasten kann die Schreibmarke um jeweils eine Stelle nach links oder rechts verschoben werden. Im XML-Layout sind sie als Spezialtasten hinterlegt und werden auf der Tastatur mit Pfeilsymbolen dargestellt, wie auf den verschiedenen Abbildungen der Tastatur in diesem Kapitel sichtbar (Abb. 3.10 ff.).

Studie 2

Mit der zweiten Studie sollte empirisch untersucht werden, wie die Verwendung der beliebtesten Kombination aus Studie 1 (vgl. 3.5.5) im Vergleich zu der Tastatur ohne Hover Detection bzw. Assistenzsysteme abschneiden würde. Die Studie wurde gemeinsam von Christoph Möhle und dem Autor durchgeführt.

Teilnehmer Die Studie wurde mit Teilnehmern des PC-Kurs für Senioren am *Familienquartierszentrums Neue Vahr Nord (FQZ)* in Bremen durchgeführt. Dies sind regelmäßige, wöchentliche Treffen bei denen interessierte Senioren mit ihrem eigenen Gerät oder einem gestellten Notebook bei der Nutzung eines Computer unterstützt werden. Nach einer kurzen Vorstellung der Testleiter und der geplanten Studie erklärten sich zunächst sieben der Teilnehmer zur Teilnahme bereit. Die Alterspanne reichte von 69 bis 82 Jahren mit einem Durchschnittsalter von 75. Fünf der Teilnehmer waren weiblich, zwei männlich. Alle Testpersonen trugen im Alltag und während des Experiments eine Sehhilfe. Eine Teilnehmerin brach das Experiment bereits nach dem Ausfüllen eines Fragebogens ab, so dass die Studie mit sechs Personen durchgeführt wurde.

Apparat Für die Studie benutzte jeder Versuchsleiter ein Samsung Galaxy S4, auf dem das *HoverZoom-Keyboard* mit den Änderungen aus Studie 1 installiert war. Die Tastatur wurde entweder so konfiguriert, dass keinerlei Assistenzfunktion aktiv waren, oder mit den folgenden Einstellungen verwendet:

- abgesetzte Vergrößerung mit kleinem Fisheye-Effekt
- Hervorhebung der Taste unter dem Finger
- Vibrationsfeedback aktiv
- Tastenstabilisierung aktiv

Vorgehen Die Durchführung der Studie fand aufgrund der eingeschränkten verfügbaren Räumlichkeiten im gleichen Zimmer statt wie der gleichzeitig stattfindende PC-Kurs. Die Versuchsleiter zogen sich mit den Teilnehmern jedoch auf separierte Plätze zurück. Nach Ausfüllen eines Fragebogens zu demographischen und allgemeinen Angaben wurden die Probanden gebeten, zwei verschiedene Texte abzuschreiben, die in der Anwendung angezeigt wurden. Dabei wurde einmal die oben beschriebene Unterstützung aktiviert, beim anderen Versuch war keine Hilfe eingeschaltet. Alle Tastendrücke wurden von der Tastatur mit Zeitstempel in eine Protokoll-Datei geschrieben. Nach der Eingabe eines Textes wurden die Teilnehmer jeweils gebeten, durch Ausfüllen eines *SUS* ihren Eindruck von der Nutzung festzuhalten. Zudem bestand die Möglichkeit, über offene Fragen Feedback zum Prototypen zu geben.

Design Bei der Studie handelte es sich um ein 2×1-*within-subjects*-Design mit der unabhängigen Variable *Hover Detection mit Assistenzsystem* mit den Leveln *aktiv* und *inaktiv*. Potentiellen Lern- und Ermüdungseffekte wurden durch abwechselnde Nutzung der Hover Detection zu Beginn des Experiments entgegengewirkt. Als abhängige Variablen wurden die Eingaben der Probanden sowie die *SUS*-Werte erfasst. Aus den Eingaben ließen sich anschließend *keystrokes per character* (KSPC), Textdistanz und Dauer berechnen.

Ergebnisse Die SUS-Scores betrugen für die Tastatur ohne Hover Detection und grafisches Feedback im Mittel 54 bei einer Standardabweichung von 21, was nach Bangor, Kortum und Miller (2008) einer Usability entspricht, die gerade so „in Ordnung“ ist. Die Werte mit Hover Detection waren sogar noch schlechter bei einem Mittelwert von 40 und einer Standardabweichung von 25. In Worten ausgedrückt ist dies ein „armseliges“ Ergebnis (ebd.). Diese niedrigen SUS-Scores drücken klar aus, dass die Usability der Tastatur für diese Zielgruppe und diesen Anwendungsfall nicht geeignet ist. Dies spiegelt sich auch in den erfassten Logdaten. Nur drei Teilnehmer schafften es überhaupt, beide Texte komplett einzugeben. Eine statistische Auswertung ist mit so einer geringen Datenmenge nicht sinnvoll. Zwei Probanden brauchten mit Hover Detection

länger als ohne bei vergleichbarer Fehlerzahl, allerdings stieg die Anzahl der pro Buchstaben notwendigen Tastendrücke. Die dritte Testperson benötigte in beiden Varianten deutlich länger als die anderen Teilnehmer, konnte sich bei der Zuschaltung der Hover Detection allerdings etwas beschleunigen. Fehlerzahl und KSPC stellten mit Abstand die höchsten Werte dar.

Aus den Beobachtungen während des Experiments ließ sich festhalten, dass die Verwendung der Hover Detection an sich kein größeres Problem darstellte. Nur eine Testperson konnte ihren Finger nicht im Annäherungsbereich halten. Vier der Probanden konnten den Finger nicht zuverlässig auf den hervorgehobenen Buchstaben absenken. Auch die Korrektur von Fehlern stellte bei vier Teilnehmern eine Hürde dar.

Alle Testpersonen äußerten bei den abschließenden Fragen den Wunsch nach größeren Tasten. Vibrations-Feedback wurde von vier Teilnehmern positiv erwähnt, das farbige Hervorheben der aktuellen Tasten von zwei. Die abgesetzte Vergrößerung empfanden vier Teilnehmer als störend. In abschließenden Gesprächen mit den Teilnehmern äußerten diese, dass ihnen sowohl Tasten als auch Textausgabe auf Smartphones generell zu klein seien.

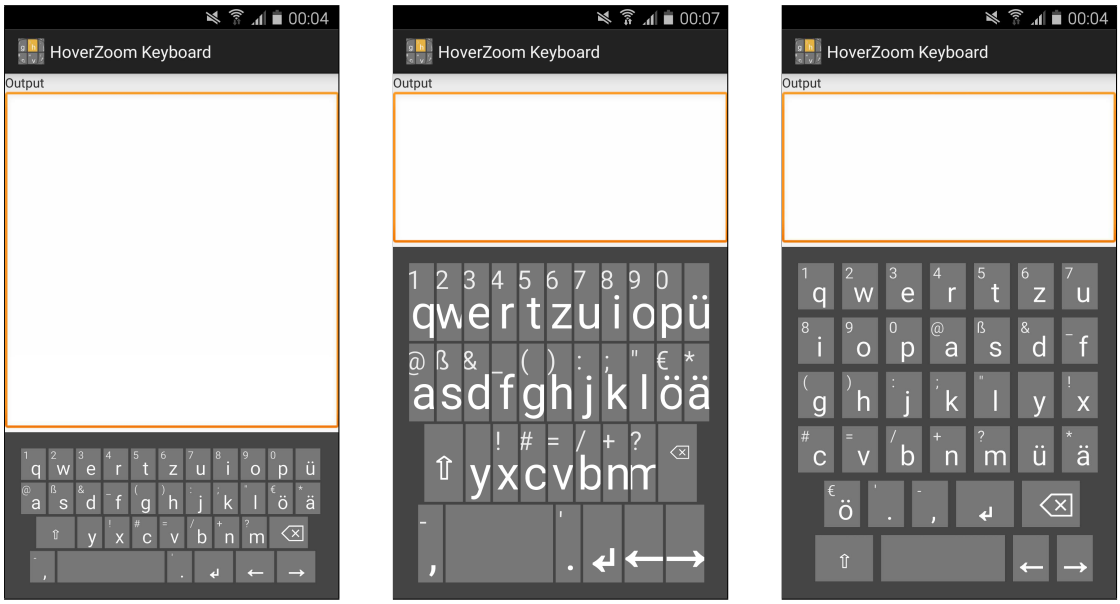
Diskussion Aus diesem Versuch einer empirischen Studie wurde sehr schnell klar, dass die gewählte Konfiguration für die hier angetroffene Zielgruppe nicht geeignet war. Daher wurde beschlossen, auf den Wunsch der Teilnehmer einzugehen und deutlich größere Tasten zu implementieren. Diese sollten bei einem weiteren Besuch im Familien- und Quartierszentrum getestet werden.

Neue Funktionen Um größere Tasten zu ermöglichen, wurde die Darstellungshöhe der Tastatur auf dem Bildschirm konfigurierbar gemacht. Zur besseren Ausnutzung des nun verfügbaren Platzes wurden zudem neue Tastaturlayouts entwickelt.

Studie 3

Nachdem in den vorherigen Studien die kontextbezogene Vergrößerung der Tasten keine positiven Einflüsse auf die Eingabeleistung der Probanden festgestellt werden konnte, sollte nun untersucht werden, ob das Problem bei die Nutzung des Smartphones an sich oder bei der Umsetzung der adaptiven Vergrößerung lag.

Dazu wurde eine Funktion implementiert, die der Tastatur mehr Platz auf dem Bildschirm einräumte. Bei ersten informellen Experimenten zeigte sich, dass eine sinnvolle Nutzung des nun verfügbaren zusätzlichen Platzes nicht mit den vorher verwendeten, auf QWERTZ basierenden Tastaturlayouts zu erreichen wäre. Da die Tastatur nur in der Höhe vergrößert wurde, aufgrund der bereits vorher kompletten Ausnutzung der Breite des Bildschirms nicht noch breiter werden konnte, wären ohne Änderung am



(a) QWERTZ-Layout mit vier Zeilen und normaler Höhe

(b) QWERTZ-Layout mit vier Zeilen und vergrößerter Höhe

(c) *manyrow*-Layout mit sechs Zeilen und vergrößerter Höhe

Abbildung 3.15: Normalhohes QWERTZ-Layout im Vergleich zu QWERTZ- und *manyrow*-Layout bei einer Höhe von 380 dp.

Layout die Tasten nur deutlich höher und dadurch relativ schmal angezeigt worden (vgl. Abb. 3.15a und 3.15b).

Mit der Entwicklung eines Tastatur-Layouts, welche die Buchstaben auf mehr Zeilen als üblich darstellten, konnte dieses Problem behoben werden (vgl. Abb. 3.15c). Die Tasten der Tastatur wurden nun auf sechs statt der üblichen vier Zielen auf der Tastatur verteilt, so dass in jeder Zeilen weniger Tasten untergebracht werden müssen. Gleichzeitig wurde der für die Tastatur zur Verfügung stehende Raum von 180 dp auf 380 dp erhöht, so dass die Größe einer individuellen Taste von 81×103 Pixel (27,68 mm^2) auf 117×151 Pixel (58,61 mm^2) anwuchs, das entspricht einer Zunahme von 112%.

Die Verwendung von mehr Zeilen als üblich hat allerdings zur Folge, dass dieses als „manyrow“ bezeichnete Layout nicht mehr der gewohnten Tastenanordnung entspricht. Inwieweit dies einen Einfluss auf die Ergebnisse haben würde, konnte nicht klar abgeschätzt werden. Auf der einen Seite war nach den vorhergehenden Untersuchungen klar, dass die Teilnehmer entweder keine großen Vorkenntnisse über das QWERTZ-Layout hatten oder diese nicht auf die Nutzung eines Smartphones übertragen konnten. Daher wurde bei der Planung des Experiments wie von Norman und Fisher (1982) beschrieben angenommen, dass die Anordnung der Zeichen auf der Tastatur ohne Vorwissen keinen großen Einfluss auf die Texteingabeleistung haben würde. Auf der anderen Seite konnten so selbst die Teilnehmer, die mit dem QWERTZ-Layout vertraut

waren, nicht mehr auf dieses Vorwissen zurück greifen. Insgesamt erwarteten wir, dass diese Nachteile durch die deutlich größere Trefferzone der einzelnen Buchstaben mehr als ausgeglichen werden würde und sich die Leistung im Vergleich zu den vorher in Studie 2 erfassten Werten und zu einer „normalgroßen“ Tastatur ohne Vergrößerungseffekte verbessern würde.

Teilnehmer Die neue Version der Tastatur wurde mit vier Teilnehmern während des PC-Seniorenclub am FQZ in Bremen sowie mit zwei Personen aus dem persönlichen Umfeld von Christoph Möhle untersucht. Alle Teilnehmer hatten bereits in den vorherigen Studien Erfahrungen mit der Tastatur sammeln können.

Apparat Die Studie wurde wie zuvor mit zwei Samsung Galaxy S4 durchgeführt, auf denen der Tastatur-Prototyp installiert war. Die Tastatur konnte für verschiedene Darstellungsgrößen konfiguriert werden. Zur Definition der Tastatur-Höhe auf dem Bildschirm wurde die abstrakte Einheit *density-independent pixel* (*dp*) verwendet. Die bisher verwendete Höhe der Tastatur von 180 dp wurde als Basishöhe verwendet; zusätzlich wurde die Tastatur mit einer Höhe von 380 dp und dem dafür entwickelten Layout *manyrow* untersucht.

Vorgehen Wie in Studie 2 wurde auch diese Untersuchung zunächst während der regulären Treffen des PC-Seniorenclubs zusammen mit Christoph Möhle durchgeführt. Die vier Teilnehmer wurden gebeten, einen Fragebogen mit persönlichen Angaben auszufüllen. Anschließend schrieben sie zwei kurze Texte ab. Einer der Texte wurde mit der normal hohen Tastatur mit QWERTZ-Layout eingegeben, der andere Text mit der hohen Tastatur mit dem *manyrow*-Layout. Die Vorlagentexte wurden bei dieser Studie ausgedruckt präsentiert, da bei der Verwendung der hohen Tastatur nicht mehr genug Platz zum Einblenden der Vorlagentexte in der verwendeten App vorhanden war. Die Hälfte der Teilnehmer begann mit der kleinen Tastatur, die andere Hälfte mit der großen Tastatur, um Lerneffekte auszugleichen. Alle berührten Tasten wurden wie zuvor in einer Logdatei mit Zeitstempel für die spätere Auswertung festgehalten. Zusätzlich zu den Durchläufen im FQZ wurde das Experiment von Christoph Möhle noch mit zwei weiteren Personen durchgeführt.

Design Das Experiment war ein 1×2-*within-subjects*-Design mit der unabhängigen Variable *Tastatur-Art*, welche die beiden Level *normales QWERTZ-Layout* und *großes manyrow-Layout* annehmen konnte. Potentiellen Lern- und Ermüdungseffekte wurden durch abwechselnde Startlevel der unabhängigen Variable entgegen gewirkt. Als abhängige Variablen wurden die Eingaben der Probanden sowie die SUS-Scores erfasst. Aus den Log-Daten ließen sich anschließend KSPC, Textdistanz und Dauer berechnen.

	QWERTZ bei 180 dp	*manyrow* bei 380 dp
KSPC	$1,48 \pm 0,32$	$1,15 \pm 0,14$
Textdistanz	$5,4 \pm 2,9$	$2,5 \pm 1,3$
Dauer	356 s ± 191 s	7 s ± 83 s

Tabelle 3.5: Vergleich von QWERTZ- und *manyrow*-Layout

Ergebnisse Ein Proband führte das Experiment nur mit der großen Tastatur durch, da die kleine für ihn nicht nutzbar war.

Die SUS-Scores betrugen für die normalgroße QUERTZ-Tastatur im Mittel 50 mit einer Standardabweichung von 21, die Nutzbarkeit der Tastatur variierte individuell stark. Bei der großen *manyrow*-Tastatur hingegen betrug der Mittelwert der *SUS*-Scores 85 mit einer Standardabweichung von 13,5. Nur einer der Probanden bevorzugte die Nutzung der kleinen Tastatur, alle anderen kamen mit der großen Tastatur besser zurecht, auch wenn sie eine ungewohnte Tastenanordnung hatte.

Bei der Erfassung der Eingabe-Performance verhinderte technische Probleme leider in drei Fällen, dass die Werte mitgeschrieben werden konnten. Aus den übrigen fünf Durchläufen ergeben sich die in Tabelle 3.5 abgedruckten Werte.

Unterschiede ließen sich auch bei der Korrektur von Fehlern beobachten. Bei Verwendung der großen Tastatur konnten alle Teilnehmer Fehler ohne Probleme ausbessern, bei der Verwendung des kleinen Layouts hatte die Hälfte der Versuchspersonen damit Schwierigkeiten.

Diskussion Sowohl die Bewertung der Usability anhand des *System Usability Score* als auch die gemessenen Performance-Werte zeigen, dass große Tasten für die Versuchspersonen deutlich besser zu verwenden waren als kleine Tasten. Damit bestätigt sich die auf dem existierenden Forschungsstand basierende Erwartung, dass besonders ältere Menschen von großen Bedienelementen profitieren und dass die Standardgröße der Tasten einer Bildschirmtastatur zu gering ist für diese Zielgruppe. Ob die Verwendung eines ungewohnten Tastaturlayouts negative Effekte auf die Eingabeleistung hatte, lässt sich aus den erfassten Daten nicht eindeutig beantworten. Sofern solche Effekte jedoch vorhanden waren, war ihre Auswirkung geringer als die Vorteile durch die besser benutzbaren großen Tasten.

Studie 4

Nachdem in den vorhergehenden Studien festgestellt wurde, dass die untersuchten Visualisierungen und Vergrößerungseffekte für ältere Menschen keinen Vorteil bei der

Nutzung einer Bildschirmtastatur brachten, sollte nun der Gegentest durchgeführt werden: Wurden die negativen Effekte durch altersbedingte Einschränkungen der Teilnehmer verursacht oder sind sie eine inhärente Eigenschaft der verwendeten kontextsensitiven Visualisierung? Sollte letzteres der Fall sein, so müssten sich die Auswirkungen auch bei der Nutzung von jungen Teilnehmern feststellen lassen. Um dies zu untersuchen wurde eine Texteingabe-Studie mit jungen Erwachsenen durchgeführt.

Teilnehmer Zehn Personen zwischen 22 und 35 Jahren nahmen an der Studie teil, davon vier weiblich und sechs männlich. Alle Teilnehmer hatten Erfahrung mit der Nutzung eines Smartphones.

Apparat Wie auch in den vorherigen Studien wurde die Tastatur auf einem Samsung Galaxy S4 installiert und evaluiert. Getestet wurden die Tastatur bei einer Höhe von 180 dp (die *normale* Größe) ohne kontextsensitive Visualisierung, mit Vergrößerung auf der Tastatur und mit abgesetzter Vergrößerung.

Vorgehen Die Teilnehmer wurden von Christoph Möhle an ihrem Wohnort aufgesucht und füllten zunächst einen allgemeinen Fragebogen aus. Nach Durchführung der drei Durchgänge des Experiments wurde jeweils ein SUS-Fragebogen ausgefüllt sowie zusätzliche Fragen schriftlich beantwortet. Wie zuvor wurden alle mit der Tastatur getätigten Eingaben für eine spätere Auswertung protokolliert.

Design Es handelte sich um ein 1×3-Design mit der unabhängigen Variable *Visualisierung* mit den drei Leveln *ohne*, *auf der Tastatur* und *abgesetzt*. Abhängige Variablen waren die *SUS*-Scores und die Texteingabeleistung.

Ergebnisse Für die Darstellung ohne kontextsensitive Visualisierung wurde im Durchschnitt ein *SUS*-Wert von 84,0 vergeben mit einer Standardabweichung von 10,1. Mit vergrößerter Darstellung direkt auf der Tastatur wurde nur ein Wert von 56,0 mit einer Standardabweichung von 19,4 erzielt. Die Variante mit abgesetzter Vergrößerung wurde durchschnittlich mit 86,8 bewertet bei einer Standardabweichung von 14,9. Diese Werte zeigen, dass eine auf der Tastaturfläche selbst dargestellte Vergrößerung die Usability deutlich beeinträchtigt. Eine räumlich versetzte Vergrößerung hingegen ist in etwa so gut nutzbar wie eine Tastatur ohne zusätzliche kontextsensitive Vergrößerung.

Aus den protokollierten Einträgen wurden *keystrokes per character* (KSPC), Textdistanz und Dauer der Eingabe berechnet. Die Durchschnittswerte sind in Tabelle 3.6 aufgeführt.

Visualisierung	ohne	auf der Tastatur	abgesetzt
KSPC	$1,22 \pm 0,03$	$1,21 \pm 0,14$	$1,26 \pm 0,20$
Textdistanz	$3,8 \pm 1,9$	$3,7 \pm 1,2$	$4,0 \pm 1,4$
Dauer	$125\,\text{s} \pm 31\,\text{s}$	$209\,\text{s} \pm 41\,\text{s}$	$122\,\text{s} \pm 30\,\text{s}$

Tabelle 3.6: Texteingabeleistung junger Nutzer

Anhand des zusätzlichen Fragebogens konnten weitere Informationen gewonnen werden. Während Vibrationsfeedback, Tastengröße oder Pfeiltasten neutral oder sogar positiv bewertet wurden, fiel das Urteil über die Darstellung von Vergrößerungseffekten direkt auf der Tastatur durchweg negativ aus. Den Versuchspersonen fielen dabei vor allem die *Unruhe* auf der Tastatur, die Verzerrung des dargestellten Buchstabens sowie die unpräzise Steuerung auf.

Diskussion Aus den Antworten und Messwerten der Studie mit jungen Nutzern kann der Schluss gezogen werden, dass eine vergrößerte Darstellung direkt auf der Tastatur nicht funktioniert, unabhängig vom Alter der jeweiligen Nutzergruppe. Als Ursache lässt sich die Verzögerung zwischen Bewegung des Fingers und Reaktion der Vergrößerung identifizieren. Ob diese Art Visualisierung besser bewertet würde, wenn die Latenz deutlich geringer wäre, kann nicht abschließend beantwortet werden. Eine nach oben versetzte Darstellung kann hilfreich sein, wurde von den Nutzern in dieser Studie zumeist ignoriert und beeinflusste die Nutzung der Tastatur dadurch zumindest nicht negativ. Die geringe Größe der Tasten scheint bei jüngeren Nutzern kein großes Problem darzustellen. Zusätzliche Hilfen wie Pfeiltasten oder Vibration zur Bestätigung eines Tastendrucks werden positiv angenommen.

Einordnung der Ergebnisse

In der Diplomarbeit von Christoph Möhle wurde untersucht, ob ältere Nutzer von Vergrößerungen unter einem schwebenden Finger bei der Nutzung einer Bildschirmtastatur profitieren können. In auf einander aufbauenden Studien stellte sich heraus, dass eine größere Tastendarstellung tatsächlich eine deutlich bessere Akzeptanz und Eingabeperformance unter den Nutzern hervorrief. Dabei ist es allerdings wichtig, dass die Tasten dauerhaft größer sind und nicht mittels einer Bildschirmlupe oder ähnlichen Funktionen nur Bereiche der Tastatur vergrößert werden. Eine mögliche Erklärung dafür ist, dass die hier verwendete auf Hover Detection basierende kontextspezifische Vergrößerung als *hektisch* wahrgenommen wurde. Neben den für Vergrößerungen auf begrenztem Raum spezifischen Effekten (vgl. Abschnitt 2.4.1) spielt hier sicherlich auch die bemerkbar hohe Latenz der Hover Detection eine Rolle (vgl. Abschnitt 3.3.1). Außerdem wurde festgestellt, dass das fehlende haptische Feedback bei Nutzung einer Bildschirmtastatur durch Verwendung eines Vibrationsfeedbacks bei Betätigung einer Taste zumindest teilweise ausgeglichen werden kann.

Eine potentielle Zielgruppe für kontextbasierte Vergrößerung einer Bildschirmtastatur könnten die 40–50-Jährigen sein, die sich bereits an die Nutzung von Smartphones gewöhnt haben, nun im Verlauf des Alterungsprozesses aber zunehmenden Einschränkungen wie beginnender Altersweitsichtigkeit unterworfen sind. Eine selektive Vergrößerung könnte helfen, diese negativen Effekte auszugleichen und ein gleichbleibend gutes Nutzungserlebnis zu ermöglichen. Nachdem in der letzten, mit jungen Erwachsenen ohne Einschränkungen durchgeführten Studie festgestellt werden konnte, dass die untersuchte Technik keine positiven Resultate erzielt, wurde auf weitere Untersuchungen mit dieser zusätzlichen Zielgruppen verzichtet.

3.5.6 Untersuchung mit sehbehinderten Menschen

Die Nutzung moderner Kommunikationsmittel ist für Sehbehinderte und Blinde sehr wichtig, um trotz ihrer Einschränkung eine gleichwertige Teilhabe am gesellschaftlichen Leben erreichen zu können. Dazu gehört auch, die Verschiebung von gesprochener Kommunikation zu schriftlicher Kommunikation mittels Textnachrichten mitgehen zu können. Während ältere Mobiltelefone dafür aufgrund der physikalisch vorhandenen, haptisch zu erkennenden Tasten zunächst geeigneter erscheinen, ermöglichen sie keine Nutzung internetbasierter Kommunikationsdienste wie *Facebook* oder *Whatsapp*.

Alle modernen Smartphones bieten jedoch umfangreiche Assistenzfunktionen, um eine Nutzung durch diese Zielgruppe zu erleichtern. Diese basieren auf einer Kombination von akustischem Feedback und einem zweistufigen Nutzungsprozess des Touchscreens: Der Nutzer kann den Bildschirminhalt zunächst untersuchen, ohne Aktionen auszulösen. Mittels Sprachsynthese werden dabei mit dem Finger berührte Elemente vorgelesen oder ihre Funktion beschrieben. Zum Betätigen einer Schaltfläche ist eine weitere Interaktion notwendig, etwa die Berührung mit einem zusätzlichen Finger oder ein doppeltes Berühren der Schaltfläche. Dies funktioniert prinzipiell zwar gut genug um eine tägliche Nutzung zu ermöglichen, ist aufgrund des mehrstufigen Interaktionskonzepts teilweise jedoch umständlich.

In der vom Autor betreuten Diplomarbeit von Block (2015) wurde daher untersucht, ob stattdessen die Verwendung von Hover Detection auf Smartphones für blinde oder sehbehinderte Nutzer eine Alternative darstellen könnte[4]. Der Fokus lag dabei wie zuvor aufgrund der sehr kleinen Tasten einer Bildschirmtastatur auf der Texteingabe.

Aufgrund der sehr eingeschränkten potentiellen Anzahl von Nutzern wurde von vornherein davon ausgegangen, dass eine quantitative statistische Untersuchung nicht möglich sein würde. Daher wurden Interviews mit Mitgliedern der Zielgruppe geführt.

[4] Die Erweiterungen des *HoverZoom-Keyboard* sowie die Durchführung der Interviews wurden von Adrian Block vorgenommen. Die Ergebnisse werden hier kurz und paraphrasiert vorgestellt; für eine detaillierte Wiedergabe der Ergebnisse sei jedoch auf die Diplomarbeit an sich verwiesen.

Neue Funktionen

Für die Untersuchung der Eignung von Audiofeedback auf Hover Detection wurde das *HoverZoom-Keyboard* um eine Sprachausgabe erweitert. Diese gibt dem Nutzer Feedback zu den Steuerelementen, die sich gerade unter dem Finger befinden. Als erste umgesetzte Funktion wurde das Vorlesen der Buchstaben umgesetzt.

Für die Implementierung wurde der bereits in Android vorhandene Sprachausgabedienst verwendet. Dieser bietet neben der Angabe der zu erzeugenden Laute auch die Möglichkeit, Eigenschaften wie Tonhöhe oder Geschwindigkeit der synthetisierten Sprache zu beeinflussen.

Ausgabe von Tasten Das zur Definition des Tastaturlayouts verwendete XML-Format wurde um zusätzliche Attribute für die Sprachausgabe erweitert. Mit ihrer Hilfe ist es möglich, die beim Überstreichen und Betätigen einzelner Tasten auszugebenden Laute und Worte zu beeinflussen. Dies wird für einige Tasten verwendet, die keinen einzelnen Buchstaben repräsentieren, sondern eine Sonderfunktion der Tastatur darstellen, wie etwa die Umschalttaste. Zusätzlich können die Attribute auch verwendet werden, um die erzeugte Sprache zu verbessern, indem unnatürlich klingende Syntheseergebnisse lautmalerisch beschrieben werden.

Um dem Nutzer sowohl für die Erkundung der Tastatur mittels Hover Detection als auch für die Betätigung der Tasten eine hörbare, aber dennoch unterscheidbare Rückmeldung zu geben, wurden verschiedene Tonhöhen bei der Ausgabe verwendet. Eine Taste, die sich unter dem Finger befindet, erzeugt eine Ausgabe mit höherer Frequenz als eine Berührung derselben Taste. Zusätzlich wurde auch eine Funktion implementiert, die statt einer Sprachausgabe einfach nur ein kurzes Klickgeräusch ausgibt, wenn der Finger über einer neuen Taste erkannt wird.

Vorlesen von vollständigen Worten Für zusätzliche Kontrolle wurde zudem eine Funktion eingebaut, die bei Betätigung der Leertaste das bis dahin eingegebene Wort einmal komplett vorliest. Mit ihrer Hilfe kann überprüft werden, ob bei der Eingabe Fehler unterlaufen sind.

Integration in *HoverZoom-Keyboard* Die neuen Funktionen wurden so in die bestehende Tastatur integriert, dass alle Features einzeln über die Einstellungen der Anwendung aktiviert oder deaktiviert werden konnten.

Erstes Interview BSVB

Der Blinden- und Sehbehindertenverein Bremen (BSVB) ist eine Selbsthilfevereinigung für Blinde und Sehbehinderte in Bremen. Er bietet „Beratung und Betreuung der blinden und sehbehinderten Menschen sowie von Blindheit Bedrohten und deren Angehörige“ sowie „Kooperation mit allen Einrichtungen des Blindenwesens und solchen Institutionen, die die Interessen blinder und sehbehinderter Menschen unterstützen“ (Lämmle 2016).

Um Feedback direkt von Betroffenen zu erhalten, bot sich eine Zusammenarbeit mit dem Verein an. Nach einer Anfrage über einen Email-Verteiler des Vereins stellten sich zwei Mitglieder für ein gemeinsames Interview zur Verfügung. Einer der Teilnehmer (T1) nutzte privat Android-Geräte, während bei dem anderen Interview-Partner (T2) Vorkenntnisse bei der Nutzung eines iPhone existierten. T1 besaß noch eine Restsehfähigkeit von 30%, womit eine Smartphone-Nutzung noch ohne Verwendung der Sprachfunktionen möglich war. Das private Gerät war ein Samsung Galaxy S4, daher waren für die Nutzung des Prototypen Vorkenntnisse mit dem Betriebssystem Android vorhanden. Bei T2 betrug die vorhandene Sehstärke nur 5%, sie galt damit als blind. Nach Eigenaussage war ihr nur noch die Unterscheidung von Hell-Dunkel-Kontrasten möglich. Hier diente ein Apple iPhone 5 als privat genutztes Smartphone, das hauptsächlich unter Verwendung der sprachbasierten Assistenzfunktionen bedient wurde.

In dem Interview wurden ein Galaxy S5 mit *HoverZoom-Keyboard* sowie ein iPad der ersten Generation verwendet. Das *HoverZoom-Keyboard* verfügte bei diesem Interview über erste Funktionen zum Erzeugen von Audiofeedback, sodass der jeweils unter dem schwebenden Finger befindliche Buchstabe vorgelesen wurde. Bei Berührung einer Taste wurde der ausgegebene Buchstabe ebenfalls noch einmal vorgelesen. Nach einer kurzen Einführung in die Bedienung eines Android-Smartphones wurde T2 gebeten, mit dem *HoverZoom-Keyboard* kurze Texte einzugeben. Sie wurde instruiert, etwaige Gedanken oder Probleme laut zu äußern, zudem wurde ihr Verhalten beobachtet. Im Anschluss demonstrierte sie noch, wie sie ihr vorhandenes Gerät nutzte. Das Vorgehen mit T1 ähnlich, jedoch konnte aufgrund der vorhandenen Erfahrungen mit der Nutzung eines Android-Smartphones auf die Einführung verzichtet werden.

Ergebnisse Die Tastatur konnte von T1 gut bedient werden. Es reichte allerdings die optische Vergrößerung der Tastatur aus, die zusätzliche Sprachausgabe wurde als nicht notwendig empfohlen. Die Funktion wurde dennoch als sinngemäß „interessant und für Leute mit abnehmender bzw. schlechterer Sehkraft eine möglicherweise sinnvolle Unterstützung“ eingeschätzt. Besonderes Merkmal der Nutzung des Smartphones durch T2 war, dass nach Erfassen eines Ziels auf dem Bildschirm sehr schnelle, „hektische“ Finger-Bewegungen vorgenommen wurden, die zu schnell für die Hovererkennung waren. Diese schnellen Bewegungen waren auch eine häufige Fehlerquelle bei der Berührung des anvisierten Ziels, da sie zu Lasten der Präzision gingen. Außerdem kam es öfter zu unabsichtlichen Berührungen des Bildschirms. Da die Sprachausgabe

beim Schweben über einer Taste und bei der Berührung einer Taste nicht unterscheidbar waren, wurden diese unabsichtlichen Berührungen nicht erkannt. Bei beiden Teilnehmer fiel auf, dass die Hover Detection in den Randbereichen des Bildschirms nur unzuverlässig funktionierte. Beim Audiofeedback hingegen war noch nicht verständlich genug, welche Aktion das Vorlesen ausgelöst hatte. Außerdem führte die Sprachausgabe zu unwillkommenen Verzögerungen bei der Bedienung des Geräts. Daher wurde die Funktion überarbeitet, so dass sich die Ausgabe für eine Taste unter dem Finger und eine tatsächlich ausgelöste Taste in der Tonhöhe unterschieden. Generell wurde die Sprachausgabe beschleunigt und unnötige Bestandteile entfernt, um eine zeitliche Straffung des Audiofeedback zu erreichen. Zur Kontrolle des bereits geschriebenen Textes wurde eine Funktion integriert, die das Texteingabefeld vorliest wenn es berührt wird.

Interview Georg-Droste-Schule

Die „Georg-Droste-Schule für Sehen und visuelle Wahrnehmung“ in Bremen ist eine Blinden- und Sehbehindertenschule, die den Lehrplan der Klassenstufen 1 bis 10 abdeckt. Auf eine Anfrage von Adrian Block hin stellten sich im zwei Lehrerinnen der Schule als Interviewpartner T3 und T4 zur Verfügung. Bei T3 lag eine starke Kurzsichtigkeit vor, T4 besaß nur noch auf einem Auge 10% Restsehstärke. Beide Teilnehmerinnen hatte keine Erfahrungen mit der Nutzung eines Smartphones, stattdessen wurden Mobiltelefone mit dem klassischen 12-Tasten-Layout verwendet. Diese Geräte waren den Teilnehmerinnen so vertraut, dass sie auf ihnen Nachrichten und Texte allein auf Basis des haptischen Feedbacks eingeben konnten. Hauptsächlich wurden die Telefone jedoch für Anrufe verwendet.

Auch in diesem Interview wurde das Samsung Galaxy S5 mit installiertem *HoverZoom-Keyboard* verwendet. Die Teilnehmerinnen wurden zunächst nach ihren Gewohnheiten bei der Nutzung ihrer Mobiltelefonen befragt. Anschließend wurde das mitgebrachte Smartphone mit den vorinstallierten Eingabehilfen vorgestellt und den Teilnehmerinnen zum Kennenlernen überlassen. Nach einer Eingewöhnungsphase wurden sie gebeten, mit sowohl mit dem *HoverZoom-Keyboard* als auch der vor Samsung vorinstallierten Tastatur Text einzugeben und dabei ihre Gedanken laut zu äußern.

Ergebnisse Beide Teilnehmerinnen bevorzugten die vorinstallierte Tastatur im Querformat gegenüber der Tastatur mit Hover Detection und Audiofeedback. Die Sprachausgabe wurde entweder als zu langsam und störend oder zumindest als überflüssig wahrgenommen. Generell verließen sie sich bei der Nutzung der Geräten auf ihre vorhandenen Restsehfähigkeiten, in einem Fall (T4) unter Verwendung einer (optischen) Lupe zur Vergrößerung von Anzeigen auf dem Bildschirm. Am liebsten wäre aber beiden die Verwendung einer externen Tastatur mit ertastbaren, physikalischen Tasten.

Die Verwendung der Hover Detection war in den Randbereichen des Bildschirms problematisch. Ein Ergebnis dieses Interviews war, dass das haptische Feedback einer physikalischen Tastatur für Sehbehinderte einen echten Nutzungsvorteil darstellt. Wenn eine solche jedoch nicht zur Verfügung steht, so müssen zumindest die Tasten einer Bildschirmtastatur so groß wie möglich dargestellt werden.

Zweites Interview BSVB

Für dieses Interview stellte sich ein weiteres Mitglied des Blinden- und Sehbehindertenvereins Bremen zur Verfügung. Die Teilnehmerin (T5) verfügte zum Zeitpunkt des Interviews über eine Sehstärke von 2,5%, die aufgrund einer Erkrankung des Sehnervs stetig weiter abnimmt. Es waren keine Erfahrungen mit der Nutzung eines modernen Smartphones vorhanden, privat wurde ein Nokia N70 verwendet. Dieses Modell bietet zwar Merkmale aktueller Geräte wie die Installation zusätzlicher Anwendungen, das Bedienkonzept basiert jedoch noch auf der klassischen numerischen Tastatur mit 12 Tasten. Das Gerät wurde von der Teilnehmerin mit einer Sprachsynthese-Software ausgestattet, es waren also Erfahrungen mit der Nutzung eines sprachbasierten Assistenzsystems vorhanden. Dennoch wurde das Gerät nur sekundär für textbasierte Kommunikation verwendet.

Wie zuvor diente das *HoverZoom-Keyboard* auf einem Samsung Galaxy S5 als Demonstrationsobjekt. Um sich mit dem unbekannten Gerät vertraut zu machen, wurde der Teilnehmerin zunächst eine kurze Anleitung gegeben sowie Zeit zum Ausprobieren des Betriebssystems und der Assistenzfunktionen zu geben. Im Anschluss wurde die Tastatur mit Hover Detection vorgestellt und zum Eingeben von Texten verwendet.

Ergebnisse Auch bei diesem Interview zeigten sich erneut vor allem in den Randbereichen zu große technische Probleme, um Hover Detection zuverlässig nutzen zu können. Außerdem kam es öfter zu unbeabsichtigten Berührungen des Bildschirms. Die eingebauten Assistenzfunktion des Smartphones zur Texteingabe hingegen wurden gut angenommen.

Drittes Interview BSVB

Das letzte der Einzelinterviews fand mit einem weiteren über den BSVB kontaktieren Teilnehmer (T6) statt. Er verfügte über eine Restsehstärke von ca. 2-3% und nutzte einen Tablet-Computer mit Android als Betriebssystem, der über eine Bildschirmdiagonale von 25,4 cm und eine eingebaute Telefoniefunktion verfügte. Dank des großen Bildschirms war eine große Schriftdarstellung möglich. Zusätzlich aktivierte der Teilnehmer wenn notwendig eine Bildschirmlupe, sodass eine Nutzung des Geräts im Alltag gut möglich war. Telefongespräche konnten mit einem Bluetooth-Headset ebenfalls geführt werden.

Aufgrund der vorhandenen Erfahrung mit dem Betriebssystem Android war bei diesem Interview keine Anleitung zur Bedienung des Gerätes notwendig und es konnte gleich mit dem Prototypen des *HoverZoom-Keyboard* die Eingabe von Texten getestet werden.

Ergebnisse Die noch vorhandene Restsehstärke ermöglichte in Kombination mit der Vergrößerung der Tasten unter dem Finger eine Nutzung zur Texteingabe. Allerdings wurde im direkten Vergleich dennoch die im Gerät integrierte Assistenzfunktion zur Texteingabe bevorzugt. Auch bei diesem Teilnehmer traten am Rand des Bildschirms Probleme mit der Hover Detection auf. Wieder zeigte sich, dass die vorhandenen Hilfsfunktionen eine Bedienung des Gerätes ermöglichen. Audiofeedback zur Fingerposition half T6 nicht, eine Vergrößerung der Darstellung unter dem Finger hingegen schon.

Zusammenfassung

In fast allen der Interviews zeigte sich, dass das in der Tastatur integrierte Audiofeedback sehbehinderten Nutzern keine Vorteile brachte. Eine große Rolle spielte dabei laut den Nutzern die unzuverlässig funktionierende Hover Detection in Kombination mit dem fehlenden taktilen Feedback bei der Nutzung der Hover Detection. Existierende Assistenzfunktionen in modernen Betriebssystemen für Mobilgeräte ermöglichen jedoch auch Nutzern mit geringer oder gar keiner vorhandenen Restsehstärke die Nutzung entsprechender Geräte und lassen sie so am modernen Kommunikationsleben teilhaben.

4 Diskussion und Ausblick

Die Durchführung der Experimente zum Thema „Verwendung von Hover Detection zur Verbesserung von Texteingabe auf mobilen Geräten“ wurde geprägt von Usability-Problemen, die primär technisch bedingt waren. Sowohl die hohe Latenz als auch die Ungenauigkeit in den Randbereichen des Bildschirms sowie die geringe Ausdehnung des Tiefenbereichs, in dem die Hover Detection funktionierte, verhinderten eine intuitive und mit einer guten User Experience verbundene Nutzung.

Wie stark diese Einflüsse bei ungeübten Testpersonen waren, wurde in den ersten Experimenten mit Angehörigen der Zielgruppe „Senioren“ klar. Zwar äußerten sich die Testpersonen in der Regel erst einmal positiv über die jeweils untersuchten Prototypen, in der praktischen Nutzung hingegen wurde schnell die mangelnde Praxistauglichkeit kritisiert .

Um die Testpersonen nicht zu überfordern, wurde entgegen der ursprünglichen Planung der Schwerpunkt bei den Experimenten nicht auf die Untersuchung der Texteingabeleistung bei Verwendung des *HoverZoom-Keyboard* gelegt. Statt dessen wurde untersucht, ob und wie Hover Detection sinnvoll bei der Nutzung eines Smartphones eingesetzt werden kann, vor allem wenn der Nutzer eine einschränkte Sehfähigkeit besitzt. Dabei zeigte sich, dass diese Zielgruppe am meisten von einer vergrößerten Darstellung profitiert. Diese könnte in Verbindung mit einer gut funktionierenden Hover Detection auch kontextbasiert dargestellt werden und durchaus einen Mehrwert bieten.

Die Kombination von Audiofeedback und Hover Detection stellte sich als schwierig zu nutzen heraus. Das fehlende haptische Feedback führte hier noch stärker zu Problemen bei der Verwendung, so dass kein Nutzer dieser Variante den Vorzug gegenüber einer berührungsbasierten Eingabehilfe gab.

4.1 Wissenschaftlicher Beitrag

Mit Hover Detection bei der Nutzung von Smartphones wurde eine spezielle Form von *Around-the-device*-Interaktion untersucht.

Die durchgeführten Experimente zeigten, dass die Genauigkeit in Zeigeaufgaben mit auf Hover Detection basierendem kontextbasierten visuellen Feedback bei der Nutzung eines berührungsempfindlichen Bildschirms erhöht werden kann. Ein großer An-

wendungsbereich für solche Zeigeaufgaben ist die Eingabe von Text mittels einer Bildschirmtastatur. Bei der Untersuchung einer Tastatur mit hover-basierter Vergrößerung mit Personen, die von einer solchen Darstellung besonders profitieren sollten, waren jedoch keine positiven Effekte messbar. Die Nutzer äußerten sich über die demonstrierten Prototypen zwar meist wohlwollend, änderten diese Meinung aufgrund der unzuverlässigen Technik und hohen Latenz bei der Nutzung jedoch zum schlechteren. Damit wurden einige bereits bekannte Erkenntnisse bestärkt:

- Größere Darstellung von Text und Schaltflächen erhöht die Nutzbarkeit von berührungsbasierten Anwendungen, vor allem für ältere Menschen und solche mit visuellen Einschränkungen.
- Zu hohe Latenz wirkt sich sehr negativ auf die Nutzbarkeit interaktiver Programme aus.

In den Gesprächen mit den Teilnehmern wurde klar, dass individuell an die Bedürfnisse der jeweiligen Nutzer anpassbare Smartphones benötigt werden, um eine Nutzung trotz einer Vielzahl möglicher Einschränkungen zu ermöglichen. Vor allem im Kontext sich verschiebender Kommunikationsvorlieben sowie aktiver Teilhabe am sozialen Leben in der Gesellschaft werden solche Geräte immer stärker benötigt werden.

4.2 Kritische Würdigung des eigenen Vorgehens

> „Hinterher weiß man immer mehr.“ (Volksmund)

Im Folgenden soll rückblickend eingeschätzt werden, was bei der Untersuchung der Forschungsfrage gut war und was besser umgesetzt hätte werden können.

4.2.1 Umgang mit technischen Problemen

Bei einer Beurteilung der umgesetzten Prototypen rein nach Usability-Gesichtspunkten hätte schon nach der Bestimmung der Latenz der Hover Detection klar sein müssen, dass die Prototypen keine Lösung des untersuchten Problems der Texteingabe würden bieten können. Dennoch konnten mit Hilfe der realisierten Tastatur wertvolle Erkenntnisse gewonnen werden: Im Prinzip würden Nutzer sich auf diese Art der Unterstützung einlassen – nur einwandfrei funktionieren müsste sie. Dass dies mit der vorhandenen Technik nicht möglich war, ist der Eingabetechnik als solcher nicht anzulasten. Da jedoch (noch) keine besser funktionierende Alternative existiert und die Entwicklung einer solchen den Fokus der Arbeit deutlich aus dem Rahmen der HCI-Forschung verschoben hätte, wurde mit der vorhandenen Technik gearbeitet.

4.2.2 Statistische Auswertung

Leider ließ die geringe Anzahl der Versuchspersonen bei den meisten Experimenten keine sinnvolle quantitative Auswertung zu. Dennoch konnten Erkenntnisse erzielt werden: Wie auch von Kjeldskov u. a. (2004) beschrieben, lassen sich viele Ergebnisse durch einige Experteninterviews erzielen. Nach Nielsen (2012) reichen bei einer qualitativen Auswertung fünf befragte Nutzer aus, um den Großteil potentieller Usability-Probleme zu entdecken. Aufgrund der in den ursprünglich als Vorstudien angelegten Interviews gewonnenen Erkenntnisse wurde daher darauf verzichtet, eine groß angelegte Studie durchzuführen, da das Ergebnis einer solchen Untersuchung mit hoher Wahrscheinlichkeit negativ ausgefallen wäre und die Versuchsteilnehmer unnötig belastet hätte.

4.3 Offene Forschungsfragen

Aufgrund der bereits ausführlich beschriebenen technischen Probleme mit der Hover Detection ließ sich die eingangs gestellte Forschungsfrage „Kann mit Hilfe von Schwebeerkennung die Texteingabe bei Nutzung einer Bildschirmtastatur verbessert werden?“ nicht abschließend beantworten. Sollte der technische Fortschritt im Bereich der räumlichen Sensorik in Zukunft eine bessere Umsetzung ermöglichen, dann müsste die Fragestellung noch einmal untersucht werden. Die Vorarbeiten dazu sind mit dieser Arbeit erledigt.

Nicht umfassend untersucht wurde zudem, ob es noch andere Anwendungsbereiche gibt, die mit der heute verfügbaren Technik schon umgesetzt werden könnten. Ein mögliches Beispiel dafür wäre die Verwendung von Hover Detection bei der musterbasierten Entsperrung von Smartphones, um die sonst entstehenden Schmierspuren auf dem Bildschirm zu vermeiden und so einen Angriffsvektor auf ein gesperrtes Gerät zu vermeiden.

4.4 Ausblick

Forschung im Bereich räumliche Interaktion ist sehr spannend, vor allem wenn die Beschränkung auf die direkte Nähe zur Bildschirmoberfläche wegfällt. Dabei sind vorhersagbare, konsistente Ergebnisse und geringe Latenzen sehr wichtig. Nur so können die Vorteile von räumlicher Interaktion zum Tragen kommen: Es ist eine Skalierung der Bedienungsgesten möglich, sodass eingeschränkte Feinmotorik kein Hindernis mehr darstellen muss. Dies wird immer wichtiger, da Smartphones und Tablets klassische Computer noch mehr als bisher in eine Nische verdrängen werden. Außerdem werden sich die Bedürfnisse der Smartphone-Nutzer verschieben: Die technikaffinen Jugendlichen

von heute sind die älteren Normalnutzer von morgen, die auf ihre gewohnte Technikнutzung nicht verzichten wollen werden. Für sie müssen Programme und Geräte bei zunehmenden körperlichen Einschränkungen wie Altersweitsichtigkeit oder Störungen der Feinmotorik bedienbar bleiben. Aber bereits heute muss älteren Nutzern die Teilhabe am modernen digitalen Leben ermöglicht werden. Vor allem Anwendungen wie die jederzeit mögliche Kommunikation mit den jüngeren Familienmitgliedern verändern auch hier Nutzungsgewohnheiten und stellen Hardware und Softwareanbieter vor neue Herausforderungen. Techniken wie die hier untersuchte Hover Detection haben das Potential, Nutzungsbarrieren abzubauen und auch in einer älter werdenden Gesellschaft freie Kommunikation, Meinungsaustausch und die freie Verbreitung von Wissen zu ermöglichen.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

3M Touch Systems (2013). *Touch Technology Brief: Projected Capacitive Technology*. Techn. Ber. 501 Griffin Brook Park Drive, Methuen MA , USA: 3M Touch Systems.

Al Faraj, Khaldoun, Mustapha Mojahid und Nadine Vigouroux (2009). „BigKey: A Virtual Keyboard for Mobile Devices“. In: *Human-Computer Interaction. Ambient, Ubiquitous and Intelligent Interaction*. Hrsg. von Julie A. Jacko. Bd. 5612. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg. Kap. 1, S. 3–10. ISBN: 978-3-642-02579-2. DOI: 10.1007/978-3-642-02580-8_1.

Amos, Evan (2011). *Kinect*. URL: https://commons.wikimedia.org/wiki/File:Xbox-360-Kinect-Standalone.png\#/media/File:Xbox-360-Kinect-Standalone.png.

– (2014). *Kinect*. URL: https://en.wikipedia.org/wiki/Kinect\#/media/File:Xbox-One-Kinect.jpg.

Arif, Ahmed S. (2012). „A survey on mobile text entry handedness: How do users input text on handheld devices while nomadic?“ In: *Intelligent Human Computer Interaction (IHCI), 2012 4th International Conference on*. IEEE, S. 1–6. ISBN: 978-1-4673-4367-1. DOI: 10.1109/ihci.2012.6481818.

Atanasov, Lachezar (2008). *Nokia 6300*. URL: https://commons.wikimedia.org/wiki/File:Nokia_6300_(1).jpg.

Atchison, David A., Elizabeth J. Capper und Kellie L. McCabe (1994). „Critical subjective measurement of amplitude of accommodation.“ In: *Optometry & Vision Science* 71.11, S. 699–706.

Bangor, Aaron, Philip T. Kortum und James T. Miller (2008). „An Empirical Evaluation of the System Usability Scale“. In: *International Journal of Human-Computer Interaction* 24.6, S. 574–594. DOI: 10.1080/10447310802205776.

Baudisch, Patrick und Gerry Chu (2009). „Back-of-device Interaction Allows Creating Very Small Touch Devices“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '09. Boston, MA, USA: ACM, S. 1923–1932. ISBN: 978-1-60558-246-7. DOI: 10.1145/1518701.1518995.

Baudisch, Patrick u. a. (2002). „Keeping Things in Context: A Comparative Evaluation of Focus Plus Context Screens, Overviews, and Zooming“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '02. Minneapolis, Minnesota, USA: ACM, S. 259–266. ISBN: 1-58113-453-3. DOI: 10.1145/503376.503423.

Bederson, B. B. und A. Boltman (1999). „Does animation help users build mental maps of spatial information?“ In: *Information Visualization, 1999. (Info Vis '99) Proceedings. 1999 IEEE Symposium on*. IEEE, S. 28–35. ISBN: 0-7695-0431-0. DOI: 10.1109/infvis.1999.801854.

Block, Adrian (2015). „Hover-Erkennung mit akustischem Feedback auf Smartphones für Blinde und Sehbehinderte“. Diplomarbeit. Bibliothekstr. 1, 28359 Bremen: Universität Bremen.

Brooke, John (1996). „SUS-A quick and dirty usability scale“. In: *Usability evaluation in industry* 189, S. 194.

Butler, Alex, Shahram Izadi und Steve Hodges (2008). „SideSight: Multi-"TouchÏnteraction Around Small Devices“. In: *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology*. UIST '08. Monterey, CA, USA: ACM, S. 201–204. ISBN: 978-1-59593-975-3. DOI: 10.1145/1449715.1449746.

Carey, John (2009). *Getting in touch with capacitance sensor algorithms*. URL: http://www.embedded.com/design/prototyping-and-development/4008781/Getting-in-touch-with-capacitance-sensor-algorithms.

Carmeli, Eli, Hagar Patish und Raymond Coleman (2003). „The Aging Hand“. In: *The Journals of Gerontology Series A: Biological Sciences and Medical Sciences* 58.2, S. M146–M152. ISSN: 1758-535X. DOI: 10.1093/gerona/58.2.m146.

Cockburn, Andy, Amy Karlson und Benjamin B. Bederson (2009). „A Review of Overview+Detail, Zooming, and Focus+Context Interfaces“. In: *ACM Comput. Surv.* 41.1, S. 1–31. ISSN: 0360-0300. DOI: 10.1145/1456650.1456652.

Colgan, Alex (2014). *How Does the Leap Motion Controller Work?* URL: http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/.

David, Paul A. (1985). „Clio and the Economics of QWERTY“. In: *The American Economic Review* 75.2.

Dix, Alan u. a. (2003). *Human-Computer Interaction (3rd Edition)*. 3. Aufl. Prentice Hall. ISBN: 0130461091.

Drewes, Heiko (2010). „Only one Fitts' law formula please!“ In: *CHI '10 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '10. Atlanta, Georgia, USA: ACM, S. 2813–2822. ISBN: 978-1-60558-930-5. DOI: 10.1145/1753846.1753867.

Findlater, Leah und Joanna McGrenere (2008). „Impact of Screen Size on Performance, Awareness, and User Satisfaction with Adaptive Graphical User Interfaces“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '08. Florence, Italy: ACM, S. 1247–1256. ISBN: 978-1-60558-011-1. DOI: 10.1145/1357054.1357249.

Finstad, Kraig (2006). „The System Usability Scale and Non-native English Speakers“. In: *J. Usability Studies* 1.4, S. 185–188. ISSN: 1931-3357.

Fisk, Arthur D. u. a. (2009). *Designing for older adults: Principles and creative human factors approaches*. CRC press.

Fitts, Paul M. (1954). „The information capacity of the human motor system in controlling the amplitude of movement.“ In: *Journal of Experimental Psychology* 47.6, S. 381–391. ISSN: 0022-1015. DOI: 10.1037/h0055392.

Fleetwood, Michael D. u. a. (2002). „An Evaluation of Text-Entry in Palm OS – Graffiti and the Virtual Keyboard“. In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 46.5, S. 617–621. ISSN: 1541-9312. DOI: 10.1177/154193120204600504.

Furbyx92 (2013). *[How-To][Code] AirView Samsung Galaxy S4*. URL: http://forum.xda-developers.com/showthread.php?t=2347001.

Furnas, G. W. (1986). „Generalized fisheye views“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Bd. 17. CHI '86 4. Boston, Massachusetts, USA: ACM, S. 16–23. ISBN: 0-89791-180-6. DOI: 10.1145/22627.22342.

Google Inc. (2015). *Project Soli*. URL: https://www.google.com/atap/project-soli/.

– (2016). *Supporting Multiple Screens | Android Developers*. Google Inc. URL: https://developer.android.com/guide/practices/screens_support.html.

Greenberg, Saul und Bill Buxton (2008). „Usability Evaluation Considered Harmful (Some of the Time)“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '08. Florence, Italy: ACM, S. 111–120. ISBN: 978-1-60558-011-1. DOI: 10.1145/1357054.1357074.

Grossman, Tovi u. a. (2006). „Hover Widgets: Using the Tracking State to Extend the Capabilities of Pen-operated Devices“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '06. Montréal, Québec, Canada: ACM, S. 861–870. ISBN: 1-59593-372-7. DOI: 10.1145/1124772.1124898.

Gunawardana, Asela, Tim Paek und Christopher Meek (2010). „Usability Guided Key-target Resizing for Soft Keyboards“. In: *Proceedings of the 15th International Conference on Intelligent User Interfaces*. IUI '10. Hong Kong, China: ACM, S. 111–118. ISBN: 978-1-60558-515-4. DOI: 10.1145/1719970.1719986.

Gutwin, Carl (2002). „Improving focus targeting in interactive fisheye views“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '02. Minneapolis, Minnesota, USA: ACM, S. 267–274. ISBN: 1-58113-453-3. DOI: 10.1145/503376.503424.

Han, Jefferson Y. (2005). „Low-cost Multi-touch Sensing Through Frustrated Total Internal Reflection“. In: *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology*. UIST '05. Seattle, WA, USA: ACM, S. 115–118. ISBN: 1-59593-271-2. DOI: 10.1145/1095034.1095054.

Harrison, Chris und Anind K. Dey (2008). „Lean and Zoom: Proximity-aware User Interface and Content Magnification“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '08. Florence, Italy: ACM, S. 507–510. ISBN: 978-1-60558-011-1. DOI: 10.1145/1357054.1357135.

Hewett u. a. (1996). *ACM SIGCHI Curricula for Human-Computer Interaction*. ACM SIGCHI.

Hobday, Stephen W. (1985). „Keyboard design to fit hands and reduce postural stress“. In: *Proceedings of the 9th Congress of the International Ergonomics Association, Bournemouth, UK. London, UK: Taylor & Francis*, S. 457–8.

Informationsverarbeitung, Internationale Föderation für (2001). *Ergebnisse 43. Intersteno Kongress Hannover*. URL: http://org.intersteno.it/uploads/ClassificationsListHannover2001.pdf.

International Organization for Standardization (2010). *ISO 9241-210:2010 - Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems*. Techn. Ber.

Jacko, Julie A., Hrsg. (2012). *Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications, Third Edition (Human Factors and Ergonomics)*. 3. Aufl. CRC Press. ISBN: 1439829438.

Jin, ZhaoXia, Tom Plocher und Liana Kiff (2007). „Touch Screen User Interfaces for Older Adults: Button Size and Spacing“. In: *Universal Acess in Human Computer Interaction. Coping with Diversity*. Hrsg. von Constantine Stephanidis. Bd. 4554. Lecture Notes in Computer Science. Springer Berlin Heidelberg, S. 933–941. DOI: 10.1007/978-3-540-73279-2_104.

Jones, Brett u. a. (2012). „Around Device Interaction for Multiscale Navigation“. In: *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services*. MobileHCI '12. San Francisco, California, USA: ACM, S. 83–92. ISBN: 978-1-4503-1105-2. DOI: 10.1145/2371574.2371589.

Kaltenbrunner, Martin und Ross Bencina (2007). „reacTIVision: A Computer-vision Framework for Table-based Tangible Interaction“. In: *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*. TEI '07. Baton Rouge, Louisiana: ACM, S. 69–74. ISBN: 978-1-59593-619-6. DOI: 10.1145/1226969.1226983.

Kaplan, E. L. und Paul Meier (1958). „Nonparametric Estimation from Incomplete Observations“. In: *Journal of the American Statistical Association* 53.282, S. 457–481. DOI: 10.1080/01621459.1958.10501452.

Kersten, Mik und Gail C. Murphy (2005). „Mylar: A Degree-of-interest Model for IDEs“. In: *Proceedings of the 4th International Conference on Aspect-oriented Software Development*. AOSD '05. Chicago, Illinois: ACM, S. 159–168. ISBN: 1-59593-042-6. DOI: 10.1145/1052898.1052912.

Kester, Jill D. u. a. (2002). „Memory in elderly people“. In: *The handbook of memory disorders* 2, S. 543–567.

Kjeldskov, Jesper u. a. (2004). „Is It Worth the Hassle? Exploring the Added Value of Evaluating the Usability of Context-Aware Mobile Systems in the Field“. In: *Mobile Human-Computer Interaction - MobileHCI 2004*. Hrsg. von Stephen Brewster und Mark Dunlop. Bd. 3160. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg. Kap. 6, S. 61–73. ISBN: 978-3-540-23086-1. DOI: 10.1007/978-3-540-28637-0_6.

Klaus, Harald u. a. (2012). „Die mobile "Generation plus Anforderungen und Potenziale“. In: *Smart Mobile Apps*. Hrsg. von Stephan Verclas und Claudia Linnhoff-Popien. Xpert.press. Springer Berlin Heidelberg, S. 545–558. DOI: 10.1007/978-3-642-22259-7_36.

Kratz, Sven u. a. (2012). „PalmSpace: Continuous Around-device Gestures vs. Multitouch for 3D Rotation Tasks on Mobile Devices“. In: *Proceedings of the International Working Conference on Advanced Visual Interfaces*. AVI '12. Capri Island, Italy: ACM, S. 181–188. ISBN: 978-1-4503-1287-5. DOI: 10.1145/2254556.2254590.

Kristensson, Per O. und Shumin Zhai (2004). „SHARK2: A Large Vocabulary Shorthand Writing System for Pen-based Computers“. In: *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*. UIST '04. Santa Fe, NM, USA: ACM, S. 43–52. ISBN: 1-58113-957-8. DOI: 10.1145/1029632.1029640.

Kurniawan, Sri (2008). „Older people and mobile phones: A multi-method investigation“. In: *International Journal of Human-Computer Studies* 66.12, S. 889–901. ISSN: 10715819. DOI: 10.1016/j.ijhcs.2008.03.002.

Lämmle, Birte (2016). *Blinden- und Sehbehindertenverein Bremen*. Blinden- und Sehbehindertenverein Bremen. URL: http://www.bsvb.org/.

Lau, Daniel (2013). *Gamasutra: Daniel Lau's Blog - The Science Behind Kinects or Kinect 1.0 versus 2.0*. URL: http://www.gamasutra.com/blogs/DanielLau/20131127/205820/The_Science_Behind_Kinects_or_Kinect_10_versus_20.php.

Lazar, Jonathan, Jinjuan H. Feng und Harry Hochheiser (2010). *Research Methods in Human-Computer Interaction*. 1. Aufl. Wiley. ISBN: 0470723378.

Levenshtein, Vladimir I. (1966). „Binary codes capable of correcting deletions, insertions, and reversals“. In: *Soviet physics doklady*. Bd. 10. 8, S. 707–710.

Lewis, James R., Kathleen M. Potosnak und Regis L. Magyar (1997). „Keys and keyboards“. In: *Handbook of human-computer interaction*, S. 1285–1315.

Likert, Rensis (1932). *A technique for the measurement of attitudes*. Hrsg. von Archives of Psychology. Bd. 22. 140. The Science Press.

Lin, Min u. a. (2007). „How Do People Tap when Walking? An Empirical Investigation of Nomadic Data Entry“. In: *Int. J. Hum.-Comput. Stud.* 65.9, S. 759–769. ISSN: 1071-5819. DOI: 10.1016/j.ijhcs.2007.04.001.

Löchtefeld, Markus u. a. (2015). „Detecting Users Handedness for Ergonomic Adaptation of Mobile User Interfaces“. In: *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia*. MUM '15. Linz, Austria: ACM, S. 245–249. ISBN: 978-1-4503-3605-5. DOI: 10.1145/2836041.2836066.

Lohmann, Kris (2013). *System Usability Scale (SUS) - An Improved German Translation of the Questionnaire | Minds*. URL: http://minds.coremedia.com/2013/09/18/sus-scale-an-improved-german-translation-questionnaire/.

MacKenzie, I. Scott (2013). *Human-Computer Interaction: An Empirical Research Perspective*. Burlington, USA: Morgan Kaufmann. ISBN: 9780124058651.

MacKenzie, I. Scott und R. William Soukoreff (2002). „Text Entry for Mobile Computing: Models and Methods,Theory and Practice“. In: *Human-Computer Interaction* 17.2-3, S. 147–198. DOI: 10.1080/07370024.2002.9667313.

– (2003). „Phrase Sets for Evaluating Text Entry Techniques“. In: *CHI '03 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '03. Ft. Lauderdale, Florida, USA: ACM, S. 754–755. ISBN: 1-58113-637-4. DOI: 10.1145/765891.765971.

MacKenzie, I. Scott und Kumiko Tanaka-Ishii (2010). *Text Entry Systems: Mobility, Accessibility, Universality (Morgan Kaufmann Series in Interactive Technologies)*. 1. Aufl. Morgan Kaufmann.

MacKenzie, I. Scott und Colin Ware (1993). „Lag As a Determinant of Human Performance in Interactive Systems“. In: *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*. CHI '93. Amsterdam, The Netherlands: ACM, S. 488–493. ISBN: 0-89791-575-5. DOI: 10.1145/169059.169431.

Martin, Benoit u. a. (2009). „Performance of finger-operated soft keyboard with and without offset zoom on the pressed key“. In: *Proceedings of the 6th International Confe-*

rence on Mobile Technology, Application & Systems. Mobility '09. Nice, France: ACM. ISBN: 978-1-60558-536-9. DOI: 10.1145/1710035.1710094.

Mayzner, Mark S. und Margaret E. Tresselt (1965). „Tables of single-letter and digram frequency counts for various word-length and letter-position combinations." In: *Psychonomic monograph supplements*.

McLaughlin, Anne C., Wendy A. Rogers und Arthur D. Fisk (2009). „Using Direct and Indirect Input Devices: Attention Demands and Age-related Differences". In: *ACM Trans. Comput.-Hum. Interact.* 16.1. ISSN: 1073-0516. DOI: 10.1145/1502800.1502802.

Meier, Reto (2008). *Professional Android Application Development (Wrox Programmer to Programmer)*. 1. Aufl. Wrox. ISBN: 0470344717.

Microsoft Inc. (2016). *Richtlinien für die Zielbestimmung*. URL: https://msdn.microsoft.com/windows/uwp/input-and-devices/guidelines-for-targeting.

Migatron Corp. (2016). *RPS-412A | High Accuracy Sensor | Migatron*. URL: http://www.migatron.com/high-accuracy-sensor/.

Miller, Robert B. (1968). „Response Time in Man-computer Conversational Transactions". In: *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*. AFIPS '68 (Fall, part I). San Francisco, California: ACM, S. 267–277. DOI: 10.1145/1476589.1476628.

Möhle, Christoph (2015). „Apps für ältere Menschen: Kann Annäherungserkennung bei Touchscreens verwendet werden, um älteren Menschen den Zugang zu Smartphones zu erleichtern?" Diplomarbeit. Universität Bremen.

Murata, Atsuo und Hirokazu Iwase (2005). „Usability of Touch-Panel Interfaces for Older Adults". In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* 47.4, S. 767–776. ISSN: 1547-8181. DOI: 10.1518/001872005775570952.

Nielsen, Jakob (2012). „How many test users in a usability study". In: *Nielsen Norman Group* 4.06.

Norman, Donald A. und Diane Fisher (1982). „Why Alphabetic Keyboards Are Not Easy to Use: Keyboard Layout Doesn't Much Matter". In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* 24.5, S. 509–519. ISSN: 1547-8181. DOI: 10.1177/001872088202400502.

Nuance (2017). *T9 Text Input: The Global Standard for Mobile Text Input | Nuance*. Nuance Communications, Inc. URL: http://www.nuance.com/mobile/mobile-solutions/text-input-solutions/t9.html.

Oulasvirta, Antti u. a. (2005). „Interaction in 4-second Bursts: The Fragmented Nature of Attentional Resources in Mobile HCI". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '05. Portland, Oregon, USA: ACM, S. 919–928. ISBN: 1-58113-998-5. DOI: 10.1145/1054972.1055101.

Parhi, Pekka, Amy K. Karlson und Benjamin B. Bederson (2006). „Target Size Study for One-handed Thumb Use on Small Touchscreen Devices". In: *Proceedings of the 8th Conference on Human-computer Interaction with Mobile Devices and Services*. MobileHCI '06. Helsinki, Finland: ACM, S. 203–210. ISBN: 1-59593-390-5. DOI: 10.1145/1152215.1152260.

Park, Yong S. und Sung H. Han (2010). „Touch key design for one-handed thumb interaction with a mobile phone: Effects of touch key size and touch key location".

In: *International Journal of Industrial Ergonomics* 40.1, S. 68–76. ISSN: 01698141. DOI: 10.1016/j.ergon.2009.08.002.

Plaisant, C., D. Carr und B. Shneiderman (1995). „Image-browser taxonomy and guidelines for designers“. In: *IEEE Software* 12.2, S. 21–32. ISSN: 0740-7459. DOI: 10.1109/52.368260.

Pollmann, Frederic, Dirk Wenig und Rainer Malaka (2014). „HoverZoom: Making On-screen Keyboards More Accessible“. In: *CHI '14 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '14. Toronto, Ontario, Canada: ACM, S. 1261–1266. ISBN: 978-1-4503-2474-8. DOI: 10.1145/2559206.2581173.

Rauschenbach, Uwe, Stefan Jeschke und Heidrun Schumann (2001). „General rectangular fisheye views for 2D graphics“. In: *Computers & Graphics* 25.4, S. 609–617. ISSN: 00978493. DOI: 10.1016/s0097-8493(01)00089-9.

Raynal, Mathieu und Philippe Truillet (2007). „Fisheye Keyboard: Whole Keyboard Displayed on PDA“. In: *Human-Computer Interaction. Interaction Platforms and Techniques*. Hrsg. von JulieA Jacko. Bd. 4551. Lecture Notes in Computer Science. Springer Berlin Heidelberg, S. 452–459. DOI: 10.1007/978-3-540-73107-8_51.

Sauer, Jürgen, Katrin Seibel und Bruno Rüttinger (2010). „The influence of user expertise and prototype fidelity in usability tests“. In: *Applied Ergonomics* 41.1, S. 130–140. ISSN: 00036870. DOI: 10.1016/j.apergo.2009.06.003.

Schieber, Frank (2003). „Human factors and aging: Identifying and compensating for age-related deficits in sensory and cognitive function“. In: *Impact of technology on successful aging*, S. 42–84.

Shannon, C. E. (2001). „A Mathematical Theory of Communication“. In: *SIGMOBILE Mob. Comput. Commun. Rev.* 5.1, S. 3–55. ISSN: 1559-1662. DOI: 10.1145/584091.584093.

Shneiderman, B. (1991). „Touch screens now offer compelling uses“. In: *IEEE Software* 8.2, S. 93–94. ISSN: 0740-7459. DOI: 10.1109/52.73754.

Siek, KatieA, Yvonne Rogers und KayH Connelly (2005). „Fat Finger Worries: How Older and Younger Users Physically Interact with PDAs“. In: *Human-Computer Interaction - INTERACT 2005*. Hrsg. von MariaFrancesca Costabile und Fabio Paternò. Bd. 3585. Lecture Notes in Computer Science. Springer Berlin Heidelberg, S. 267–280. DOI: 10.1007/11555261_24.

Sony Mobile (2012). *Floating touch*. URL: http://developer.sonymobile.com/knowledge-base/technologies/floating-touch/.

Spence, Robert und Mark Apperley (1982). „Data base navigation: an office environment for the professional“. In: *Behaviour & Information Technology* 1.1, S. 43–54. DOI: 10.1080/01449298208914435.

Synaptics Inc. (2001). *Synaptics TouchPad Interfacing Guide*. URL: http://www.synaptics.com/sites/default/files/ACF126.pdf.

Teather, Robert J. u. a. (2009). „Effects of tracking technology, latency, and spatial jitter on object movement“. In: *3D User Interfaces, 2009. 3DUI 2009. IEEE Symposium on*. IEEE, S. 43–50. ISBN: 978-1-4244-3965-2. DOI: 10.1109/3dui.2009.4811204.

Teichert, Jens u. a. (2010). „Advancing Large Interactive Surfaces for Use in the Real World“. In: *Advances in Human-Computer Interaction* 2010, S. 1–10. ISSN: 1687-5893. DOI: 10.1155/2010/657937.

Thoma, Jörg (2012). *Test Android 4.1: Butter bei die Jelly Beans - Golem.de*. URL: http://www.golem.de/news/test-android-4-1-butter-bei-die-jelly-beans-1207-93271.html.

Virzi, Robert A., Jeffrey L. Sokolov und Demetrios Karis (1996). „Usability Problem Identification Using Both Low- and High-fidelity Prototypes“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '96. Vancouver, British Columbia, Canada: ACM, S. 236–243. ISBN: 0-89791-777-4. DOI: 10.1145/238386.238516.

Vogel, Daniel und Patrick Baudisch (2007). „Shift: A Technique for Operating Pen-based Interfaces Using Touch“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '07. San Jose, California, USA: ACM, S. 657–666. ISBN: 978-1-59593-593-9. DOI: 10.1145/1240624.1240727.

Walker, Neff, David A. Philbin und Arthur D. Fisk (1997). „Age-Related Differences in Movement Control: Adjusting Submovement Structure To Optimize Performance“. In: *The Journals of Gerontology Series B: Psychological Sciences and Social Sciences* 52B.1, P40–P53. ISSN: 1758-5368. DOI: 10.1093/geronb/52b.1.p40.

White-Chu, E. Foy und Madhuri Reddy (2011). „Dry skin in the elderly: Complexities of a common problem“. In: *Clinics in Dermatology* 29.1, S. 37–42. ISSN: 0738081X. DOI: 10.1016/j.clindermatol.2010.07.005.

Wigdor, Daniel u. a. (2007). „Lucid Touch: A See-through Mobile Device“. In: *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*. UIST '07. Newport, Rhode Island, USA: ACM, S. 269–278. ISBN: 978-1-59593-679-0. DOI: 10.1145/1294211.1294259.

Wolf, Jan-Hendrik (2014). „Unterstützung der Touchinteraktion durch Annäherungserkennung auf Smartphones“. Magisterarb. Universität Bremen.

Zhai, Shumin und Per O. Kristensson (2012). „The Word-gesture Keyboard: Reimagining Keyboard Interaction“. In: *Commun. ACM* 55.9, S. 91–101. ISSN: 0001-0782. DOI: 10.1145/2330667.2330689.

A Implementierungsdetails

A.1 Verwendung von Hover-Events in Android

Um in einer Android-Anwendung Hover-Events von *AirView* empfangen und verarbeiten zu können, muss folgendes erledigt werden (Furbyx92 2013):

1. Einen Intent-Filter für die von *AirView* erzeugten Hover-Events im *activity*-Abschnitt der Datei `AndroidManifest.xml` anlegen:

```
<manifest [...]>
  <uses-sdk android:minSdkVersion="14" [...] />
  <application [...]>
    <activity [...]>
      <intent-filter>
        <action android:name="com.sec.android.airview.HOVER" />
      </intent-filter>
    </activity>
  </application>
</manifest>
```

Listing A.1: Ausschnitt aus `AndroidManifest.xml`

2. Einen Hover-Listener implementieren und auf der passenden View setzen:

```
View v = findViewById(R.id.someview);
v.setOnHoverListener(new View.OnHoverListener() {
  @Override
  public boolean onHover(View v, MotionEvent ev) {
    int x = ev.getX(); // x-Koordinate auf dem Bildschirm
    int y = ev.getY(); // y-Koordinate auf dem Bildschirm

    // hier dann Koordinaten im Shader aktualisieren

    return false;
   }});
```

Listing A.2: Hover-Listener in Android

Die vom Event gelieferten x- und y-Koordinaten (Zeile 5 und 6, Listing A.2) entsprechen dabei den Bildschirmkoordinaten des Pixels, über dem das Event ausgelöst wurde.

A.2 Grafische Darstellung von Fisheye-Effekten mit OpenGL ES

Um das visuelle Feedback für die Hover-Events zu implementieren, wurde die Schnittstelle *Open Graphics Library* in der Version für eingebettete Systeme (abgekürzt: *OpenGL ES*) verwendet. Sie erlaubt ein hardwarebeschleunigtes Berechnen von 2D- und 3D-Szenen, so dass auch bei anspruchsvollen Effekten eine flüssigen Animation möglich ist. Die Prinzipien hinter OpenGL zu erklären, wäre deutlich zu umfangreich für diese Arbeit. Zudem existieren schon sehr umfangreiche und gute Werte, die einen Einstieg in die Verwendung dieser API vermöglichen. Der Schwerpunkt hier liegt nur darauf, die Realisierung der Fisheye-Darstellung zu erläutern.

Die Tastatur wird in Android mit einer Komponente *GLView* realisiert. Für die Darstellung des Fisheye wird zunächst ein Abbild der Tastatur mit den doppelten Abmessungen der GLView erzeugt. Sie wird als Textur einem die GLView komplett ausfüllenden Quad zugewiesen. Aufgrund der automatischen Texturskalierung von OpenGL wird die Textur dabei automatisch soweit verkleinert, dass sie komplett dargestellt werden kann.

Um nun den Fisheye-Effekt zu erzeugen, benötigen wir dessen Radius r, die Position p sowie eine Funktion, die den Verlauf der Vergrößerung beschreibt. Diese Informationen stehen einem Fragmentshader zur Verfügung, der für jedes Fragment f den Vektor zwischen dessen Koordinaten und der Fingerposition berechnet. Sofern die Länge dieses Vektors kleiner ist als r, ist das Fragment Teil des Fisheye. Für die Vergrößerung wird der Vektor in Abhängigkeit der Entfernung zwischen Finger- und Fragmentposition mit einem von der Skalierungsfunktion bestimmten Faktor $s \in [0,5; 1]$ multipliziert. Dies erzeugt eine virtuelle Fragmentposition f', die sich zwischen der ursprünglichen Position f und dem Mittelpunkt p befindet. Ein Texturzugriff für f' ergibt nun die Farbe für f. Im Ergebnis erhalten alle Fragmente innerhalb von r die Farbe eines Fragments, das sich näher an p befindet, was genau einem Vergrößerungseffekt entspricht.

Durch eine Veränderung der Skalierungsfunktion können unterschiedliche Vergrößerungseffekte erzielt werden (vgl. Abb A.1). Eine monoton ansteigende Funktion ergibt einen Fisheye-Effekt, während eine Sprungfunktion mit konstanten Wertebereichen eine lineare Vergrößerung ergibt.

,

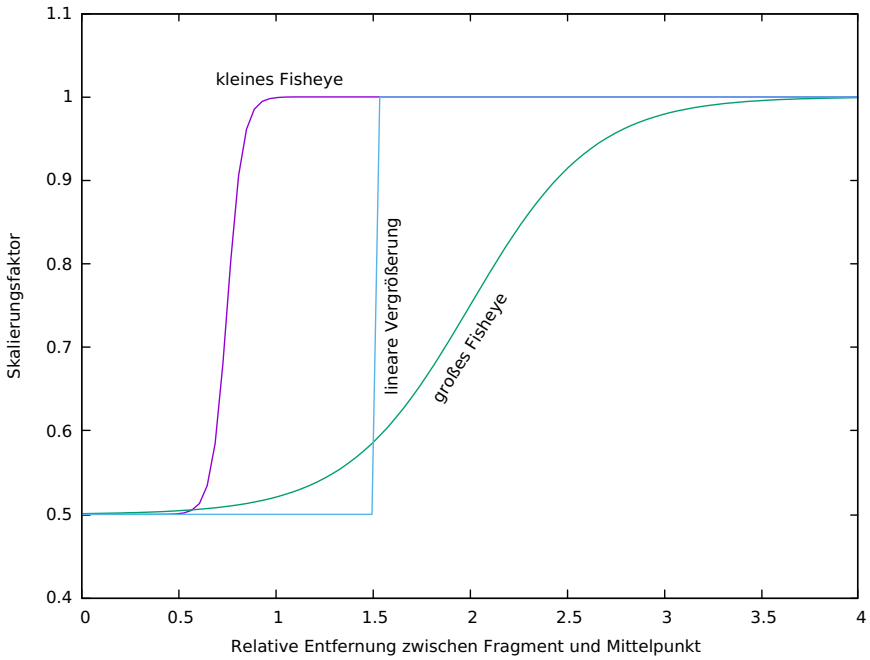


Abbildung A.1: Skalierungsfunktionen der verschiedenen Vergrößerungen